PRESERVING PRIVACY IN SOCIAL NETWORKING SYSTEMS: POLICY-BASED CONTROL AND ANONYMITY

by

Amirreza Masoumzadeh

Master's in Computer Engineering (Software), Sharif University of Technology, 2007

Bachelor's in Computer Engineering (Software), Ferdowsi University of Mashhad, 2004

Submitted to the Graduate Faculty of

the School of Information Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2014

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCES

This dissertation was presented

by

Amirreza Masoumzadeh

on

Aug 1st 2014

PRESERVING PRIVACY IN SOCIAL NETWORKING SYSTEMS: POLICY-BASED CONTROL AND ANONYMITY

Amirreza Masoumzadeh, PhD

University of Pittsburgh, 2014

Social Networking Systems (SNSs), such as Facebook, are complex information systems involving a huge number of active entities that provide and consume enormous amounts of information. Such information can be mainly attributed to the users of SNSs and hence, can be considered privacy-sensitive. Therefore, in contrast to traditional systems where access control is governed by system policies, enabling individual users to specify their privacy control policies becomes a natural requirement. The intricate semantic relationships among data objects, users, and between data objects and users further add to the complexity of privacy control needs. Moreover, there is immense interest in studying social network data that is collected by SNSs for various research purposes. Anonymization is a solution to preserve user privacy in this case. However, anonymizing social network datasets effectively and efficiently is a much more challenging task than anonymizing tabular datasets due to the connectedness of the users in a social network graph.

In this dissertation, we propose approaches and methods that facilitate preserving user privacy in terms of providing both fine-grained control of information and utility-preserving anonymization. In particular, we propose an ontology-based privacy control framework that enables fine-grained specification and enforcement of privacy control policies by both users and SNS providers. Our framework allows an SNS provider to determine privacy control policy authorities for SNS information, and allows users to specify advanced policies, that in addition to fine-grained policy specification, enables sharing of authority over protected resources. Based on such an ontology-based foundation, we also propose a framework to support novel privacy policy analysis tasks in SNSs. Furthermore, we propose a framework to enhance anonymization algorithms for social network datasets in terms of preserving their structural properties without sacrificing privacy requirements set for the algorithms. The proposed approaches direct the behavior of anonymization algorithms based on concepts in social network theory. We evaluate our proposed methods and approaches by implementing a prototype of the privacy control framework, carrying out a policy analysis case study for a real-world SNS, and performing an extensive set of experiments on improving social network anonymization in terms of preserving data utility.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

I would like to acknowledge the help and support of many that made conducting this research and concluding my dissertation possible. I am grateful to my PhD advisor, Dr. James Joshi, for his supervision and for his encouragement to commit to research during my PhD studies. I would like to thank my dissertation committee members, Dr. Elisa Bertino, Dr. Peter Brusilovsky, Dr. Prashant Krishnamurthy, and Dr. Konstantinos Pelechrinis for their invaluable feedback and support on my dissertation work. I am also grateful to the faculty that provided me great career advice and support including Dr. James Joshi, Dr. Rosta Farzan, Dr. Prashant Krishnamurthy, and Dr. Hassan Karimi.

I am thankful to my friends at the LERSAIS lab, Hassan Takabi, Saman Taghavi-Zargar, Nathalie Baracaldo, Xuelian Long, Lei Jin, Jesus Gonzales, and Yue Zhang for sharing a happy and productive working environment with me during these years. I am also lucky to have great friends at the PAWS lab who supported me and shared with me enjoyable moments at the school and beyond. I appreciate the support of great staff at iSchool, specially, Kelly Shaffer who has been a great support on numerous occasions. Last but not least, I cannot thank enough the love of my life, Shaghayegh Sahebi, for encouraging and supporting me during the toughest days of my PhD, and my parents and brothers who have always been great moral support in my life. I finally acknowledge that the research presented here has been supported by the US National Science Foundation CAREER award IIS-0545912.

1.0 INTRODUCTION

Social Networking Systems (SNSs) are increasingly becoming a major type of online applications that facilitate social interactions and information sharing among a large number of users. The scale of active entities, interactions, and digital content in these complex environments brings about new security and privacy challenges. Users constantly provide contents and information to these systems, either explicitly, such as by uploading a photo, or implicitly by leaving behind interaction traces, such as by responding to an invitation. Such information can be considered privacy sensitive since it is related to the users [30, 35, 47]. Therefore, an SNS must ensure preserving its users' privacy as it shares information with various entities through its services. Several privacy problems may arise in such an environment, such as effectively expressing privacy control policies by users, controlling information objects that have multiple stakeholders, or efficiently anonymizing published social network data. In this dissertation, we aim to provide solutions to such complex privacy issues.

We show the high-level privacy preservation mechanisms that are required in an SNS in Figure 1. We assume that an SNS uses a knowledge base to store its information and provide services to its users. The knowledge base will store all application-related information such as social network, user profiles, content objects, etc. Besides the SNS operator, we consider three categories of entities that may access the information. Users may query and update the information through interaction with the SNS's user interface. Viewing your friend's profile information or updating yours are some examples. In addition, users may authorize other application providers to access data on behalf of them (i.e., delegation) in order to receive add-on services. For instance, many SNS providers nowadays provide an app platform through which they provide third party developers access to user's data. Such accesses must be authorized by users in order to respect their privacy. Integration and data exchange with

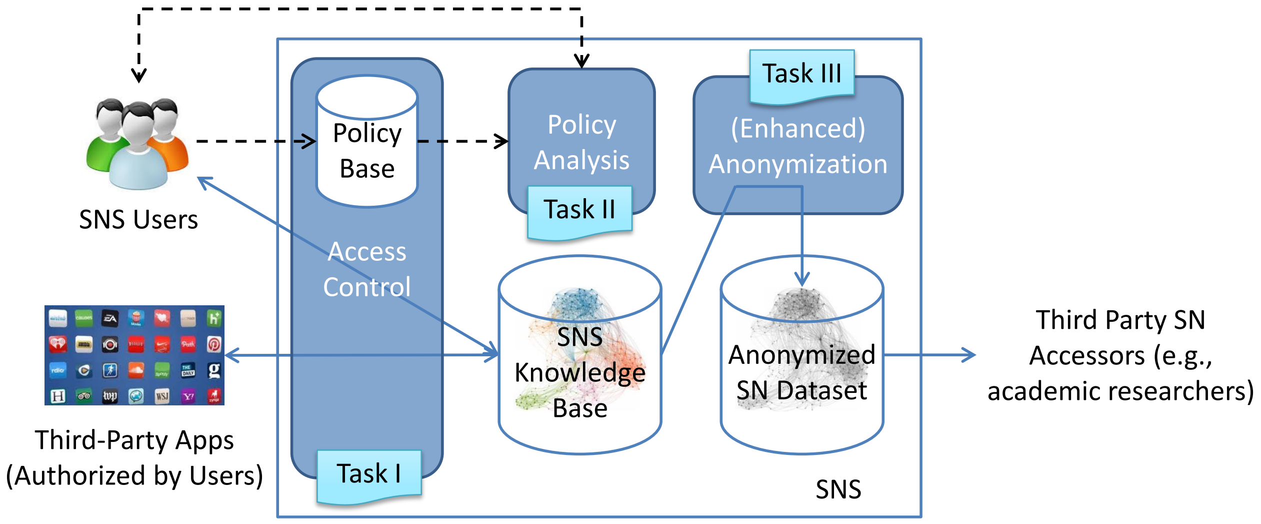


Figure 1: Overview of Privacy Preservation Requirements for SNSs

other SNSs can also be provided using a similar approach. Lastly, an SNS provider may allow third parties to access its data in an anonymized format, without requiring explicit authorization from its users. Many researchers in different domains such as academia, business, and even government, are interested in the study of social network data [61, 58, 67, 11]. The purpose of such a study is typically to investigate various structural properties and patterns both at node level and network as a whole depending on the application of interest. Market researchers, academic researchers, and advertisers are a few examples of such third parties. SNSs are by their nature a good platform to collect such information. However, since a social network dataset may carry privacy-sensitive information about its users it is important to ensure that users cannot be reidentified in the anonymized dataset.

As depicted in Figure 1, an SNS should preserve users' privacy by mediating access to SNS knowledge base based on privacy control policies set by the system and its users. Such policies will restrict access to the information related to a user by other users and applications. Moreover, a rigorous anonymization process makes sure to export anonymized data from the main knowledge base that respects users' privacy for use by third party accessors. In this dissertation proposal, we propose research tasks to develop mechanisms that addresses certain access control and anonymization challenges in SNSs, and overcome the limitations of the previously-proposed approaches in the literature.

1.1 CHALLENGES AND MOTIVATIONS

In this section, we discuss our motivations and problem statement and present challenges that need to be addressed. The volume of privacy-sensitive information in SNSs calls for tighter security measures for protection against attackers. But it also creates unique challenges and complexities to mediate accesses of the legitimate entities, as indicated in Figure 1. As per a general goal of SNSs, users are motivated to expand their social connectivity and awareness through interactions and content sharing with each other. However, as the social connections of a user grows, so does the complexity of privacy implications. The increased variety of social contents and connections requires more fine-grained control on privacy-sensitive information. Leveraging third party applications also raises similar challenges. Furthermore, as users release more privacy-sensitive information, including their connections that can help in (re)identifying them, compiling an anonymized and still useful dataset for the use by external entities becomes more challenging.

There are major issues with the current practice of privacy control policies in SNSs such as Facebook. They provide some control in the form of privacy settings to their users. However, the privacy control features provided by these systems are usually limited, and not so flexible and robust [10]. Moreover, they seem to be implemented incrementally without detailed formal modeling, a practice that has led to many controversies [71, 43, 1]. Several desirable control features are missing and there exist no way of verifying consistency in policy specification and enforcement. The following examples show some pitfalls of taking such an approach by Facebook. In Facebook, a user can choose to hide her relationship status with her significant other. But one can learn about that relationship if the significant other happens not to hide it. In other words, users cannot control disclosure of some intuitively privacy-sensitive information. As an example of inconsistency in policy enforcement, even if a user disables being publicly listed in Facebook she will be still listed in the public listings of the groups which she has joined (Facebook has disabled public listing of group members recently). Digital resources in SNSs are comprised of various data types. Also, different annotation methods such as tagging and commenting are common in these systems. These all introduce a variety of semantic relations among objects. In particular, it is important to ensure the protection of not only the basic data entities and values, but also their relations. For instance, a person tagged in a photo might not be only concerned about being tagged, but also about who else has been tagged in the same photo, and who actually owns the photo. In order to truly capture the fine-grained protection requirements in SNSs, it is important to have an appropriate data model. Most of the major approaches for access control in SNSs, such as trust-based policies [48, 49, 14, 15] and relationship-based access control [24, 25] focus on enabling expressive policies with regards to access subjects. These approaches neglect to capture fine-grained information modeling required for SNSs in order to express fine-grained privacy control policies on information objects. Moreover, unlike in traditional systems where security administrators are in charge of access control policy, in an SNS, users should be recognized as the main authority over access control policies protecting the information related to them. A flexible authority model is required to determine each user's authority over different resources. This feature has not been addressed in existing work. Some key challenges are

Challenge 1. How can SNS users be provided authority to express privacy control policies for the information related to them?

Challenge 2. How to support fine-grained specification of privacy control policy on SNS information?

Challenge 3. How can an SNS ensure consistent enforcement of privacy control policies?

Since SNSs have a complex policy structure, it is important to provide policy analysis techniques that help maintain the desired properties for policies and avoid privacy violation risks. As an example, let us discuss transparency of policies in a typical SNS such as Facebook. Users are able to control access to some of their resources through privacy settings. However, such privacy settings are by no means complete in the sense that they do not control access to all the potentially privacy-sensitive information about a user. That is the case even for Facebook, which has fairly the most extensive set of privacy settings among SNSs. For the rest of the information related to the user and not indicated in the privacy settings, access is governed by a set of fixed rules set by the SNS itself. We call such policies as *system defined policies*. The issue is that usually the system defined policies are not clearly described to the users. Therefore, users are unsure about what to expect from the system. Users need to learn about them either by harvesting help pages in the SNS or by observing the system's behavior. Worse is that, since the system defined policies are not well documented, SNSs can modify them without users noticing it and put them at great risk of privacy violations. Such issues raise the need for a formal policy analysis framework that can handle the complex privacy control policy requirements in these environments. For example, using such an analysis framework, one can test the transparency of policies in an SNS. Most of the recent related work on policy analysis (e.g., [46, 52]) focuses on XACML policies. Although XACML is an expressive policy language that can support many traditional security policies it is not suitable for representing complex privacy control policies in SNSs such as setting "*Who can see posts you have been tagged in on your timeline?*" in Facebook. Thus, the key challenge is

Challenge 4. How can we formally analyze privacy control policies in an SNS and provide useful policy analysis for system operators and users?

As previously mentioned, anonymization of social network datasets are vital if they are to be shared with third parties. A naive anonymization for social networks may simply replace real node identifiers with random ones. However, researchers have shown that such an approach is not immune from node reidentification if an adversary has certain background knowledge about network structure of a target victim [4, 32, 62]. In order to provide stronger anonymization for social networks, researchers have mainly taken two different approaches to structural anonymization. In the *graph generalization* approach [31, 81, 12], a social network is summarized in a higher-level graph, hiding details of the relationships among agents while providing overall structural summaries of the graph. In the *edge perturbation* approach [32, 80, 53, 76, 83, 33], the edge structure of the social network is modified, i.e., some edges are removed and some are added, in order to satisfy an anonymity property (typically based on k-anonymity [75]). For instance, a graph can be modified to have degree k-anonymity where for each node there are at least $k-1$ other nodes with the same degree. Therefore, an adversary that knows the degree of a victim node will not be able to reidentify her by a probability larger than $1/k$. An expected consequence of structural anonymization is that running same analysis on a social network and its anonymized version may lead to different results. In order to use a social network anonymized by a generalization method it needs to be reconstructed by randomly generating sub-structures in place of supernodes based on the reported supernode properties. Modifying links in the perturbation methods to fulfill the anonymization criteria also severely affects the network structure. One may hope that such differences are negligible, so that results are still usable. But observations show that if a social network is anonymized up to an acceptable degree, the resultant network becomes highly distorted, thus, severely affecting their utility for analysis purposes [62]. For instance, a node with a low centrality value may become one with high centrality value because of the addition of many fake adjacent links. Such a change can reduce the accuracy of centrality analysis of the network nodes. The key issue is that anonymization methods usually focus on achieving the anonymization objectives and disregard the crucial need to preserve the original structural semantics of a social network; hence, the outcome is a significant decrease in the utility of the results. In addition to researching social network anonymization concepts and algorithms to achieve them, it is crucial to study how utility of social network dataset is affected and can be preserved in an anonymization process. Here, the key challenges are:

Challenge 5. What factor should be considered in anonymization algorithms for social network datasets in order to better preserve the utility of the data?

Challenge 6. Can we devise a generic approach to preserve social network utility in anonymization process?

1.2 OBJECTIVES

Motivated by the challenges discussed in Section 1.1, we propose an ontology-based privacy control framework for SNSs and a framework for structure-preserving anonymization of social networks. Our ontology-based privacy control framework consists of an ontology-based access control model for SNSs (OSNAC) and a privacy control policy analysis approach. Our structure-preserving anonymization framework employs concepts in social network analysis theory to improve network perturbation techniques in terms of preserving network structure. The framework enhances original algorithms using concepts of structural roles and edge betweenness. With regards to these frameworks, we study the validity of the following two hypotheses in the dissertation:

- Ontology-based privacy control framework enables fine-grained specification of privacy control policies and useful policy analysis tasks for SNSs.
- Concepts in social network analysis theory can be leveraged in edge-perturbing anonymization algorithms for social networks in order to better preserve data utility while maintaining privacy requirements.

In order to precisely show the achievement of the above-mentioned, broadly-defined hypotheses, we break them down into more specific research questions. We formulate the following three research questions with regards to the first hypothesis:

Question 1. Does OSNAC enable finer-grained specification of protected resources compared to the existing access control models for SNSs?

Question 2. Does OSNAC enforce privacy control policies correctly?

Question 3. Can our ontology-based privacy control policy analysis framework enable novel and useful policy analysis tasks for SNSs?

We also formulate the following two research questions with regards to the second hypothesis:

Question 4. Do role-enhanced and edge-betweenness-enhanced versions of edge perturbing anonymization algorithms preserve data utility better than original algorithms?

Question 5. Do role-enhanced and edge-betweenness-enhanced versions of edge perturbing anonymization algorithms respect privacy requirements of the original algorithms?

1.3 ORGANIZATION

The rest of the dissertation is organized as follows. In Chapter 2, we review the closely related work in the literature to the dissertation. In Chapter 3, we propose an ontology-based privacy control framework for SNSs, demonstrate its applicability in controlling users' privacy and policy analysis, and compare it against the proposed models in the literature. In Chapter 4, we propose a framework to enhance structure-preservation of edge perturbing anonymization schemes for social networks, and demonstrate its effectiveness using extensive experimental results. Finally, in Chapter 5, we summarize the contributions of the dissertation and discuss some limitations and future directions.

2.0 RELATED WORK

2.1 ACCESS CONTROL APPROACHES FOR SNSS

2.1.1 Relationship-Based Policies

The early efforts on access control models for SNSs rely on leveraging social connections as trust relationships. Those works are inspired by research developments in trust and reputation networks (e.g., [28, 29, 40, 41]). FOAF-Realm [48, 49] is one of the earliest approaches that quantifies the *knows relations* in the context of FOAF (Friend Of A Friend) ontology as a trust metric, and supports rules that control accesses of friends to resources in a social network by stating the maximum distance and minimal friendship level. Friendship level is a trust value between any two individuals that is computed based on indirect relationships in case a direct relationship is missing. Carminati et al. [14, 15] propose a conceptually similar but more complete and formal trust-based access control model. The access control rules are defined based on type of relationship, maximum distance and minimum trust value. Villegas et al. [77] propose to use a slightly different trust measure by automatically classifying nodes into zones. A general drawback of trust-based access control models is the usability issues. It can be very hard to comprehend and specify appropriate trust thresholds. Hence users may be left with even less protection compared to simple, conventional access control approaches. While these approaches focus mainly on subject specification based on distance and trust measures, we take a more abstract approach on subject specification. Trust information can be easily captured in the ontology and hence policies in our approach, independently from underlying trust computation mechanism. Our focus is instead on accurately capturing the protected information semantics using an ontology-based privacy control policy model.

Fong et al. [23, 3] propose a more formal and generic access control model for SNSs than trust-based approaches. In its policy predicates, the model accounts for network topology between an object owner and an accessor, communication history between the two entities (e.g., friendship request sent), and type of the object. Their proposed topology model allows specifying more variety of patterns than simple distance. For instance, access can be constrained to the case where the two users have certain number of mutual friends or are part of a clique. In a subsequent work, Relationship-Based Access Control (ReBAC) [24, 25] was proposed. ReBAC provides a family of policy languages based on modal logic that allows for combining simple policy predicates based on relationship types in a social network into more complex topology-based policies. Compared to its predecessors, ReBAC has a more formal approach and more generic and concrete topology-based policy language. However, the model focuses on the relationship between access subject and owner, and does not address the need for fine granularity modeling of protected resources in SNSs.

2.1.2 Semantic Web-Based Policies

Carminati et al. [13] propose a Semantic Web-based access control framework for SNSs that leverages OWL and SWRL. The authors define three types of policies, namely, access control policy, filtering policy, and admin policy. Access control policies are positive authorization rules. Filtering policies can limit someone's access to information by herself (which is not necessarily a security issue). Finally, admin policies express who are authorized to define the former policies. Although the authors outline an access control framework, lack of formal descriptions and implementation details leaves behind many ambiguities. In comparison, we propose a more detailed and semi-formal semantics for our model (See Section 3.2), and show the applicability by implementing a proof-of-concept framework. Also, our model captures the notion of individual authorities, and provide privacy control policies to protect the relations in the knowledge ontology as a more expressive and flexible alternative to entity protection. Ryutov et al. [69] propose a rule-based access control model for semantic networks, based on a constrained first order logic. The authors have implemented their model in an RDF-like framework. While the model is based on logic rules similar to our approach in

Section 3.2, the work introduces notions such as *attaching policies* and separating policy at subject and object levels without adequate elaboration and justification of their application. Similar to the previous work, they use relationships in specifying access subjects only and not protected resources. Therefore, the expressiveness of their policy in terms of protected resource is limited to the entity level.

In the area of Semantic Web, Rei [42] is a prominent policy language based on RDFS. Although Rei leverages Semantic Web languages, it mainly provides a generic framework to support different deontic concepts in the policy (i.e., permission, prohibition, obligation, and dispensation), and distributed policy management. However, in terms of specifying subjects and protection objects it uses generic conditions (attribute-value constraints), which are not specific to Semantic Web. In contrast, our approach is more focused on how to specify fine-grained policy rules on a knowledge base that is specified using OWL. There exists other access control solutions for RDF stores that are close to our work, although not in the context of SNSs. Reddivari et al. [66] propose RAP, a rule-based model and architecture. In RAP, access control policy is written using Jena framework rules, and supports both permit and prohibit predicates, similar to what our model features. Although no experimental results are reported, RAP does not seem to have an efficient access control enforcement method. For a given query to the RDF store, the result set is retrieved first. Then, the access control inference is performed separately for every triple in the result in order to decide whether to include it in the final result. Our query augmentation approach (See Section 3.2.4) enforces access control more efficiently. The use of access control predicates in the query avoids excessive overhead of access checks by leveraging the query processing mechanism itself. There are other approaches to access control on RDF stores that are comparatively less grounded [20, 19, 54], or support only a specific policy such as multi-level security [39].

2.1.3 Policy Analysis

There exist some work on analysis and comparison of access control policies in the literature outside of the scope of SNSs. Most of the recent literature on policy analysis focuses on XACML policies [46, 52]. EXAM [52] is a framework for analyzing policy similarity among

XACML policies. It can verify answer of a policy given a set of inputs, find the inputs for which multiple policies have the same decision, or have contrasting decisions. It relies on *multi-terminal binary decision diagrams* for policy similarity analysis. Kolovski et al. [46] propose an XACML policy analysis approach based on description logics. Since our ontology modeling language is also theoretically founded on Description Logics, there are similarities between this approach and ours in terms of their support for logic-based reasoning. However, the focus of both these frameworks are on formalizing complete XACML attribute-based policies. Although XACML is a powerful attribute-based access control language, it is not suitable to express the policies in the case of complex network model of information objects in SNSs as developed in our approach. For example, one cannot express a typical privacy setting in an SNS such as "*Who can see posts you've been tagged in on your timeline?*" using XACML. Furthermore, the notion of multiple authorities has not been addressed in these frameworks.

2.2 REIDENTIFICATION ATTACKS ON SOCIAL NETWORKS

Researchers have shown that it is possible to reidentify users based on the network structure in naively anonymized social networks (in which only explicit identifiers are removed). Backstrom et al. present a family of active/passive attacks that work based on uniqueness of some small random subgraphs embedded in a network [4]. Hay et al. study the extent of node reidentification based on structural information [32, 31]. They experiment on the use of three types of structural queries as adversary background knowledge on real, naively anonymized social networks and show significantly low k-anonymity for such background knowledge queries. Narayanan et al. propose a different attack approach that relies on input of an auxiliary, overlapping, probably publicly available social network without any assumption about structural background knowledge of an adversary [62]. Empirical evaluation of their approach shows that a third of users who have accounts both on Twitter and Flickr can be re-identified in the anonymous Twitter graph with a low error rate.

2.3 K-ANONYMITY APPROACHES FOR SOCIAL NETWORKS

The non-naive social network anonymization approaches in the literature can be categorized into two groups: graph generalization and graph perturbation.

2.3.1 Generalization

In ***generalization techniques***, the network is first partitioned into subgraphs. Then each subgraph is replaced by a supernode, and only some structural properties of the subgraph alongside linkage between clusters are reported. Hay et al. propose a k-anonymity-based generalization approach, where supernodes contain at least k nodes, which optimizes fitness to the original network via a maximum likelihood approach [31]. Zheleva et al. propose a generalization approach to avoid disclosure of exact nonsensitive edge structure, which could be used for predicting sensitive edges [81]. Campan et al. also follow a similar approach [12] but propose a greedy optimization solution that can be tuned to control information loss. In order to use a generalized social network for analysis purpose, one should sample a random graph in accordance with the reported generalized properties. Although such a network may maintain some local structural properties of the original network, much of high-level graph structure is lost [76], which impacts negatively the utility of results.

2.3.2 Perturbation

In ***perturbation techniques***, the network is modified to meet desired privacy requirements. This is usually carried out by adding and/or removing graph edges. Although, theoretically, perturbation can be introduced to graph nodes as well, it is not considered plausible because of adverse effects on the dataset.

Hay et al. propose a random perturbation approach, in which a sequence of m edge removals is followed by m edge additions [32]. Assuming an adversary needs to consider the set of possible worlds implied by m removals/additions, the authors reason that it could be intractable for an attacker to achieve exact identification. However, this cannot guarantee that the adversary will not succeed in (sufficiently accurate) identification of selected individuals. Ying et al. formulate the confidence of an adversary in identifying a node in a randomly perturbed network based on the degree of the target as background knowledge [80].

Liu et al. propose an edge perturbation approach that provides k-anonymity for vertices based on their degrees [53]. Initially a k-anonymous degree sequence for the graph is constructed, in which there exist at least k nodes of each degree and the total degree difference between the anonymized and the original degree sequence is minimum. Then the problem reduces to realizing a graph with the anonymized degree sequence from the original graph. They propose two different algorithms to solve it. The `Supergraph` algorithm greedily perturbs the original graph until it reaches the target anonymized degree sequence. Since such a greedy algorithm cannot guarantee an answer, a probing scheme is proposed by the authors that retries the procedure with slight modification of the degree sequence, until an anonymized graph is realized. The `Greedy-Swap` algorithm starts by constructing a random graph based on the anonymized degree sequence. It then modifies the graph to maximize its overlap with the original graph, while preserving the anonymized degree sequence.

Thompson et al. propose a k-anonymity-based two-phase clustering and perturbation approach [76]. Vertices are first clustered into groups of size of at least k, and then edges are greedily added/removed so that each vertex is anonymous to the vertices in its cluster. Anonymity is either based on the degree of a vertex or the degrees of a vertex and its neighbors, which is also used as similarity measure for forming clusters. They propose two alternative clustering algorithms for this purpose: `Bounded t-Means`, and `Union-Split`. Although their clustering approach seems promising, the proposed greedy perturbation algorithm based on clusters does not guarantee an answer. An approach such as probing in [53] seems necessary to take into account realizability of the graph based on formed clusters.

Zhou et al. propose a scheme to k-anonymize vertex neighborhoods [83], by constructing isomorphic neighborhoods. The method seems to be inefficient and with high graph distortion, since it requires recurring anonymization of node neighborhoods. He et al. propose a different neighborhood anonymization scheme [33], by making isomorphic groups of k graph partitions. Furthermore, we believe that making every k-grouped partitions isomorphic and adding back inter-partition edges in an isomorphic-preserving manner as adopted in [33] will create a very symmetric structure; considering the need for addition of about k^2 edges per original inter-partition edge, the result does not seem to maintain well its original structural properties in general.

3.0 A PRIVACY CONTROL FRAMEWORK FOR SOCIAL NETWORKING SYSTEMS

In this chapter, we present our ontology-based framework to capture, enforce, and analyze fine-grained privacy control policies in SNSs. The rest of the chapter is organized as follows. In Section 3.1, we detail our ontology modeling approach for SNS information that enables expressing fine-grained privacy-sensitive information resources. In Section 3.2, we propose our access control model, describing the policy specification and enforcement details, and evaluate it by comparing against existing approaches in the literature and presenting experimental results of our prototype implementation. In Section 3.3, we propose our formal approach to analyzing privacy control policies. We present a detailed analysis task on transparency of permission management in SNSs and demonstrate the applicability of our analysis approach by providing a case study.

3.1 MODELING FINE-GRAINED PERMISSIONS ON SNS INFORMATION

In this section, we propose an ontology-based model of SNS information using OWL [34] as our modeling language. We first discuss a generic information model for SNSs and explain in more detail a sample ontology that we have developed for Facebook. Clearly, a similar ontology can be developed for other SNSs. Then, we present our approach to capture privacy-sensitive information resources in such a model.

3.1.1 Semantic Web Languages, Notations, and Terminology

We leverage several Semantic Web languages in this work. In particular, we use Web Ontology Language (OWL) [57] to express the protected knowledge in an SNS and some access control decision information. We use Semantic Web Rule Language (SWRL) [37] to specify access control policy rules. In fact, we use a subset of this language, known as DL-safe rules because of its decidability property [60]. Moreover, we present how access control can be applied to knowledge base queries based on SPARQL Protocol and RDF Query Language (SPARQL).

OWL [34] is a W3C recommendation to express meanings and semantics, which builds on RDF/RDFS. There are three main concepts in OWL. A *class* is a collection of objects, which are also called individuals/instances of the class. A *property* is a directed binary relation (predicate). An *object property* relates instances of two classes, and a *data property* relates instances to data values (e.g., string values). A class or a property can be defined as subclass or subproperty of another. OWL also supports various operators on classes such as union, intersection, and complement and restrictions such as cardinality constraints on properties. OWL has three sublanguages that vary in their level of support for different operators/concepts. For the purpose of this work, we have chosen OWL DL as it provides semantic features adequate for expressing knowledge in an SNS, does not have intractability issues, and there exist various tools and packages that support it. As customary to XML-based languages, we use namespace prefixes to distinguish different ontologies. For instance, owl:Class represents the type *Class* in the OWL namespace. We interchangeably use function-style notation for representing OWL property instances and the Manchester syntax for OWL 2 [36]. For example, owns(alice, book1) and alice owns book1, are representing the same triple (predicate) in function-style and Manchester syntax, respectively. Here, owns is the property, alice is the *subject* of the property, and book1 is the *object* of the property. We use this terminology for describing reification of ontology properties. As for naming convention, in this document, class names start with upper case, and properties and instances start with lower case, e.g., class User and its instance alice.

SWRL [37] allows combining Horn-like rules with an OWL knowledge base, thereby enabling new knowledge reasoning tools. We encode access control policy rules using SWRL to reason on top of access decision information stored in an OWL-based knowledge base and infer access decision. SWRL rules have a very detailed syntax [37]. For the purpose of our work, we represent them simply as *antecedent* ⇒ *consequence*, where *antecedent* (body) is a conjunction of multiple predicates and *consequence* (head) is a single predicate. Predicates can be either unary or binary, representing either a class or a property, respectively. A notation such as $?x$ is used to declare variable x in the body/head of a rule, which can be bound to class instances. For example, the following rule expresses that if someone is a tenure-track faculty he/she has a PhD degree.

$$\mathsf{TenureTrackFaculty}(?x) \Rightarrow \mathsf{hasDegree}(?x,\ \text{“PhD”})$$

SPARQL [65] is a syntactically SQL-like language for querying RDF graphs via pattern matching. We augment SPARQL queries with access control predicates to automatically enforce access control policies when a query is evaluated.

3.1.2 SNS Ontology: Basic Concepts

We model SNS information as a set of users, digital objects, and data values, which are related to each other by relationships. In OWL terminology, we model the first two concepts using *classes* as abstract objects. The relationships between objects are captured using the following *object properties*: between users to define the social links (e.g., friendship, following, etc.), between digital objects (e.g., a comment that is related to a photo), or between users and digital objects (e.g., a user may own a photo). Moreover, class objects can be related to data values using *data properties*, e.g., the relationship between a photo and its binary content. We model annotations as a special class of digital objects. An annotation is an object that annotates a first object with a second object. Comments and photo tags are two common ways of annotation. For instance, a photo tag can annotate a photo with a user. As another example, a check-in in Foursquare is an annotation that annotates a venue (place) with a user, which may also be associated with a time stamp. These are the minimum concepts and relationship types that we capture for an SNS; they can be easily extended depending on the needs of an SNS.

Figure 2 depicts our proposed ontology for Facebook. Note that many details, especially restrictions, are not captured in the figure or in the following description. Entity is the root class of our ontology (a subclass of owl:Thing, which is OWL's built-in most general class). It is specialized by User and DigitalObject. There is a predefined instance in class User, i.e., me, for whom policy analysis is performed. The isFriendOf object property expresses friendship relationship between instances of User. Data properties such as hasFullname associate data values to a User. The owns object property defines a User as the owner of a DigitalObject. DigitalObject is the union of four major subclasses: Content, Wall, Event, and Annotation. Content represents an object that has data content such as a photo or a text (specified using hasContent data property). Classes Wall and Event, respectively, correspond to profile wall page and events that users can attend on Facebook. Annotation, as mentioned before, represents objects that instead of directly carrying content, annotate a DigitalObject (e.g., a wall or a photo) with an Entity (e.g., a wall post or a comment), using object properties annotates and annotatesWith, respectively. Actual forms of annotations are defined as its subclasses: Comment, UserTag, and WallPost. A Comment annotates a Commentable (a class which is the union of Photo and WallPost) with a Text. A UserTag annotates a UserTaggable (a class which is the union of Photo and Text) with a User. We refer to this ontology as *SNS Ontology (SNO)* in the rest of the document, and use namespace prefix *sn* to refer to its concepts and properties.

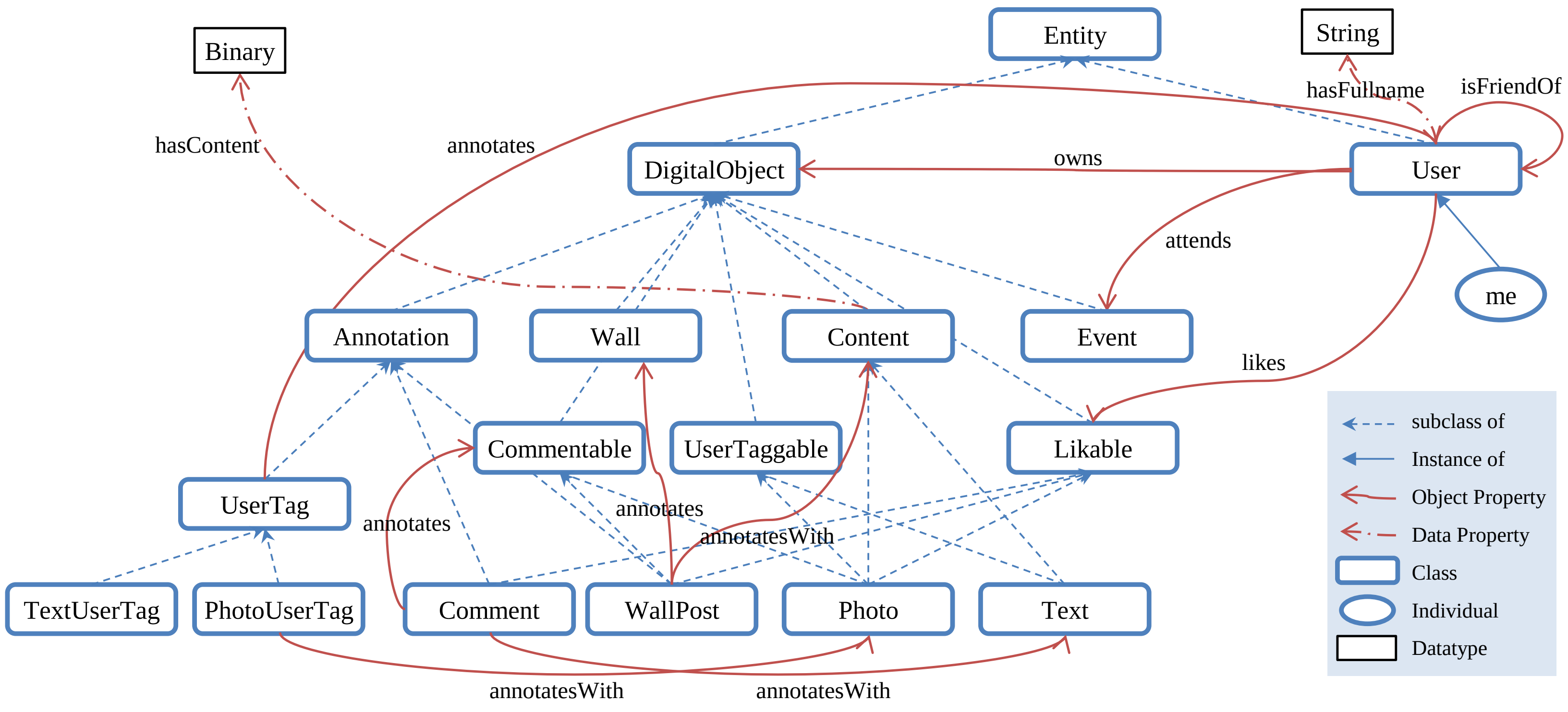


Figure 2: SNS Ontology (SNO): A Model of Concepts and Properties for Facebook

3.1.3 SNS Ontology: Restrictions

It is crucial to accurately model the data constraints in an SNS knowledge base, in order to accurately enforce access control policies and facilitate meaningful policy analysis. We employ the following restriction features in OWL to model such data constraints:

Disjoint Union of Subclasses: We ensure our class hierarchy is captured completely by defining each class as disjoint union of its subclasses, i.e., union of subclasses which are pair-wise disjoint. For instance, Annotation is equivalent to the disjoint union of Comment, UserTag, and WallPost.

Property Domain/Range: The domain and range of a property can be restricted to specific classes, e.g., owns has User and DigitalObject as its domain and range, respectively.

Property Characteristics: OWL supports several restrictions to accurately model property constraints: functional, inverse functional (i.e., the inverse of the relation is a function), transitive, symmetric, asymmetric, reflexive, and irreflexive. For instance, isFriendOf is defined as irreflexive; hence, no user can be a friend of herself.

Class Property Restriction: OWL supports existential, universal, cardinality, and value constraints for properties when applied to a class. For instance, WallPost is defined equivalent to a class that annotates exactly one Wall.

3.1.4 Running Example

In order to facilitate discussions, we provide a small scenario as a running example based on the sample ontology proposed in Section 3.1.2. We refer to this example throughout this document.

Figure 3 depicts a partial ontology that captures a typical scenario: a group of users, a photo, a comment on the photo, and tagging of users in the photo and the comment. Here, there are friendship relationships among Alice, Bob, Carol, and David. Note that Alice and Bob are related using isCloseFriendOf property, which is a subproperty of isFriendOf. Alice has a photo (photo1), in which Bob has been tagged by Carol: Carol owns photoUserTag1 which annotates photo1 with Bob. The abstract photo object is associated with its binary content using property hasContent. David has liked photo1, and has made a comment

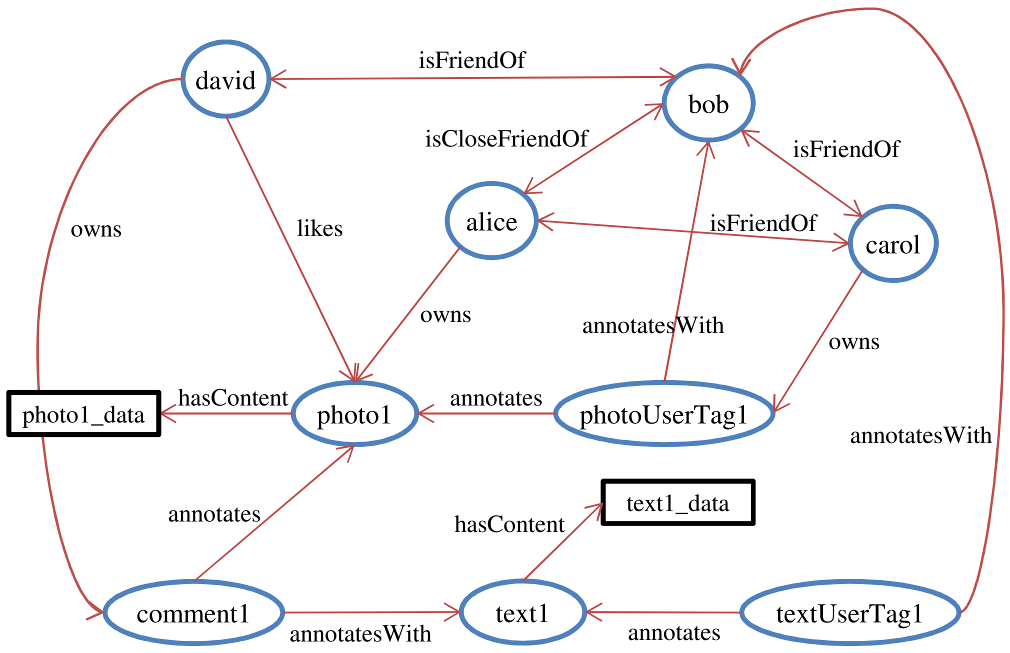


Figure 3: Running Example - A Photo Post

(comment1) on that, say mentioning how well Bob looks in the photo. comment1 annotates photo1 with a text object (text1), which is associated with its textual value using property hasContent. Moreover, by mentioning Bob in the comment, David has effectively tagged him in the comment's text: textUserTag1 annotates text1 with Bob.

3.1.5 Properties as Protected Resources

The ontology modeling approach proposed in Sections 3.1.2 and 3.1.3 captures SNS information in the forms of classes and properties. We argue that a class instance does not represent any privacy-sensitive information by itself unless its properties are considered. In other words, the essential knowledge is captured by the triples that represent properties between two instances (object properties), or between an instance and a data value (data properties). Based on this observation, we consider such triples as privacy-sensitive information that needs to be protected.

By considering the object and data properties in our running example (See Section 3.1.4) as protected resources, we can provide fine-grained access control to privacy-sensitive information. We assume a simple and effective policy authority scheme: *the owners of the endpoints of each property are eligible to define policy for that property.* Therefore, in the example, visibility of the tag is dependent on relationships annotates(photoUserTag1,photo1) (controlled by policies provided by Alice and Carol) and annotatesWith(photoUserTag1, bob) (controlled by policies provided by Bob and Carol). Carol can control disclosure of the fact that she has created the tag by controlling owns(carol, photoUserTag1). Finally, Alice can control the association of the actual binary content of the photo using relationship hasContent(photo1, photo1_data).

With regards to our proposed SNS ontology and specific to the running example, one may think that there are redundancies in tag objects and their relationships. However, such a modeling approach allows very fine granular and selective control over sensitive information. For instance, in the running example, Bob is tagged once in the photo and once in the comment. Therefore, there are two tag objects photoUserTag1 and textUserTag1 and their corresponding relationships. This allows Bob and other people involved in these tags to independently control each tag by providing separate policies for the corresponding relationships.

By considering the three basic possible actions on property instances in the ontology, i.e., selection, insertion, and deletion, different permissions can be specified. For example, the insertion of a tag can be controlled using similar policies discussed for visibility of the tags.

3.1.6 Reification of Properties

In order to specify policies, we need to define authorization characteristics about property instances in an ontology. However, OWL does not support such expressions about individual properties. In other words, there cannot be properties that can characterize other properties. We use the concept of reification to overcome this limitation. We reify each object/data property in our SNS ontology as a class in OWL. Figure 4 depicts the ontology for reified versions of properties presented in Figure 2. Class ReifiedProperty is the root of all such

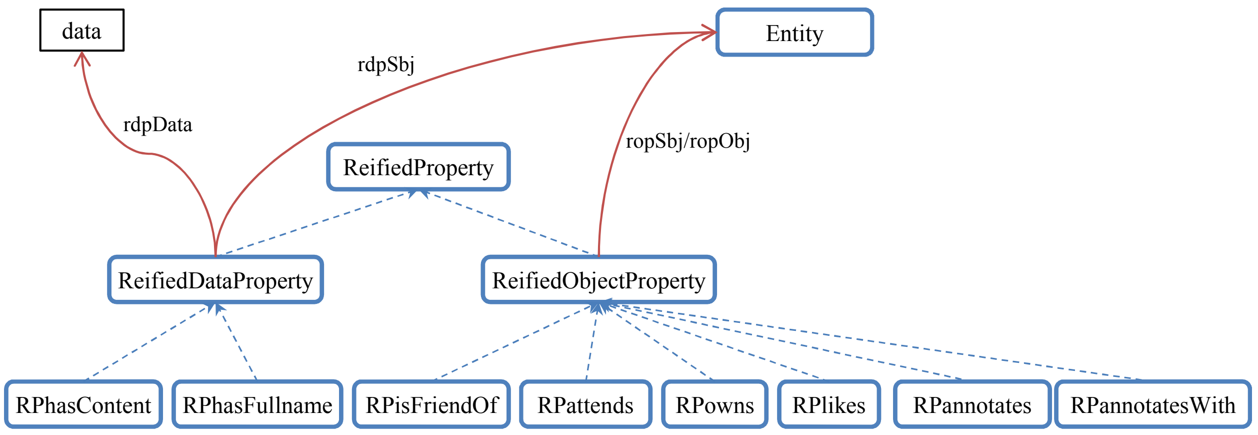


Figure 4: Representing Reified Properties in Ontology

classes, with the two subclasses ReifiedObjectProperty and ReifiedDataProperty, which are the disjoint union of the corresponding reified property classes. We define two special object properties, ropSbj and ropObj, that relate a reified object property to its subject and object, respectively. Analogously, the special object and data properties rdpSbj and rdpData relate a reified data property to its subject and data value, respectively. An example of reification of a triple from the running example is shown in Figure 5. The reification allows us to characterize instances of reified properties, such as rpAnnotates1, as the protected resources.

The reified instances of properties may be automatically generated from the current statements in the ontology. Alternatively, reified classes can replace SNS ontology properties that were described in Section 3.1.2. In the latter case, the information domain constraints considered in designing the ontology need to be enforced on the reified classes. We demonstrate

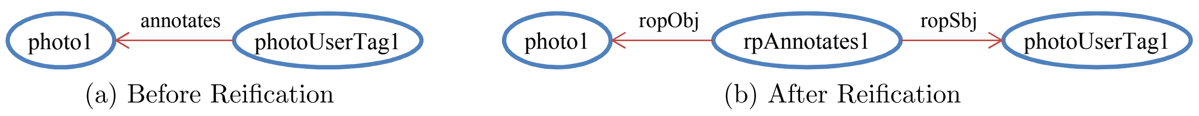


Figure 5: Reification Example: annotates Property

with an example how such a property restriction that was presented in Section 3.1.3 can be translated to be applied on reified properties. Let p be an object property in the SNS ontology, and RPp be its corresponding reified class. If p is a functional property (i.e., it assigns only one object to any subject), such a characteristic can be ensured for RPp by the following restriction: owl:Thing $\sqsubseteq$ inverse ropSbj max 1 RPp.

3.2 ONTOLOGY-BASED PRIVACY CONTROL MODEL

We propose Ontology-based Social Network Access Control (OSNAC), a rule-based access control policy model for SNSs based on Semantic Web standards. OSNAC is a fine-grained semantics-aware model that captures relationships between ontological concepts of knowledge as protected resources. For this purpose, the model relies on an ontology such as SNO (introduced in Section 3.1) that models the SNS knowledge. It also uses an access control ontology (ACO, described in Section 3.2.1) to model the policies. We assume a closed-world policy model, where an access is denied unless it is allowed according to access control policy rules.

Figure 6 shows the overall OSNAC policy framework. Access control rules are specified at two levels: user and system. At the user level, every user can express *personal authorization* rules regarding protected resources. For more flexible authorizations, users can leverage *dependent authorization*, *delegative authorization* as well as *multi-authority specification* rules. At the system level, the rules govern the overall privacy policy of the system. *Basic authority specification* rules determine which users have authority over which protected resources. They empower users by recognizing the authorizations defined at the user level as system-level authorizations. In other words, they aggregate user-level authorizations by determining the appropriate authority for protected resources. In contrast, *direct system authorization* rules indicate system authorizations that are valid independently of users' policies. Finally, conflict resolution rule(s) apply to system authorizations to determine final access decision. All policy rules, except conflict resolution, are expressed using DL-safe SWRL syntax,

In Figure 6, a higher-level policy component may be considered to aggregate its lower-

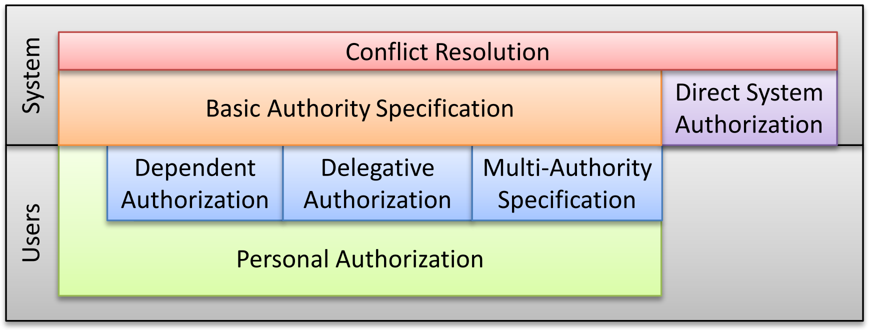


Figure 6: OSNAC Policy Framework

level policy components. For instance, basic authority specification rules aggregate user-level policies. But direct system authorizations do not aggregate any other authorizations. Similarly, conflict resolution policy aggregates system-level authorizations to determine the final access decision. We elaborate on each component of the framework in the rest of the section.

3.2.1 Policy Expression at Ontology Level

Since knowledge resources are captured in an ontology, such as SNO, the access control policies need to express them using concepts in the ontology. In order to facilitate an efficient and semantics-rich access control, we choose to capture information related to access control policy using an ontology too. For clarity, we use a separate ontology for this purpose, which we call *Access Control Ontology (ACO)*. We use namespace prefix *ac* to refer to ACO concepts and relationships, which are depicted in Figure 7. ACO is used to model and store any knowledge solely needed for access control purposes including inferences based on access control policy rules. We have already presented some part of this ontology in Section 3.1.6. We categorize the concepts and relations in ACO as follows.

- **Access Subject:** Class ac:Subject is used to specify the access subject (an instance of sn:User) of a given access request that is going to be evaluated.
- **Reified Properties:** As previously discussed in Sections 3.1.5 and 3.1.6, we consider

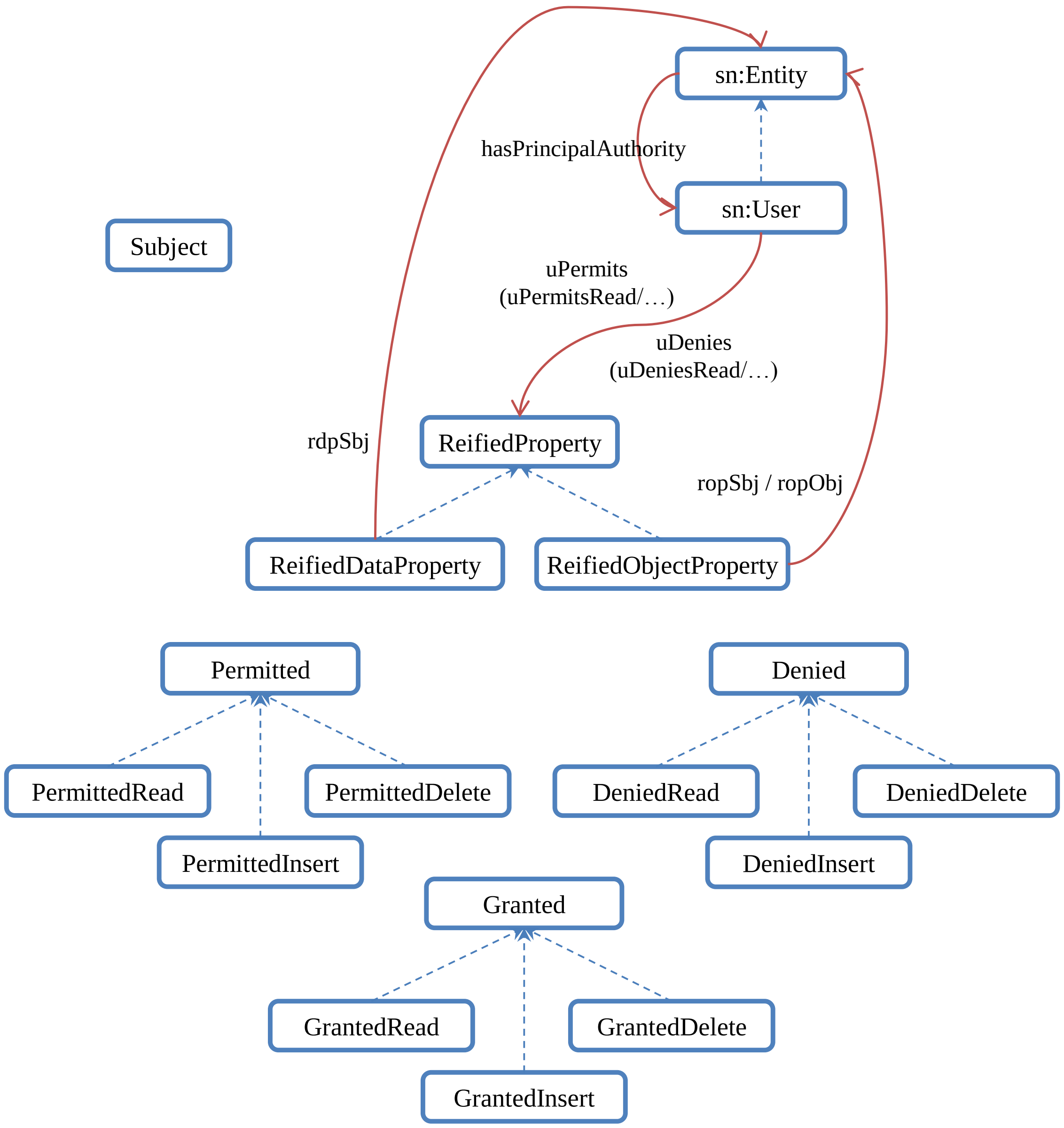


Figure 7: Access Control Ontology (ACO)

instances of SNO properties as protected resources, and represent them using reified statements. Corresponding to each property sn:x, there exists class ac:RPx, which is a subclass of ac:ReifiedProperty. Thus, a predicate such as sn:isFriendOf(Alice, Bob) in SNO is correspondingly represented in ACO using an instance of class ac:RPisFriendOf; the subject and object of the predicate are related using ac:ropSbj(Alice) and ac:ropObj(Bob), respectively.

- **User Authorizations:** User authorizations determine merely a user's decision about an access. Note that such a decision is not necessarily the effective decision that will be enforced by the SNS. Property ac:uPermits (ac:uDenies) abstractly relates the user who issues the rule to the reified property instance that is to be accessed by the subject. Depending on the mode of access, one of the descendents of ac:uPermits (ac:uDenies), i.e., either ac:uPermitsRead (ac:uDeniesRead), ac:uPermitsDelete (ac:uDeniesDelete), or ac:uPermitsInsert (ac:uDeniesInsert) is used.
- **System Authorizations:** System authorizations determine SNS policy. In most cases (where user-related protected resources are involved), such a policy may infer the authorization based on user-level authorizations. Class ac:Permitted (ac:Denied) represents an abstract notion of system-level positive (negative) authorizations, which is assigned to a reified property instance. Depending on the mode of access, one of the descendents of ac:Permitted (ac:Denied), i.e., either ac:PermittedRead (ac:DeniedRead), ac:PermittedDelete (ac:DeniedDelete), or ac:PermittedInsert (ac:DeniedInsert) is used.
- **Access Decision:** Class ac:Granted represents the final positive access decision. Similar to authorizations, depending on the mode of access, one of its descendents (ac:GrantedRead, ac:GrantedDelete, or ac:GrantedInsert) is used. A reified property class instance will be assigned to any of these classes if the corresponding access decision is resolved as positive (after resolving possible conflicts at system-level authorizations).
- **Principal Authority:** We assume there is a unique principal authority for every SNO class instance, assigned using property ac:hasPrincipalAuthority. The principal authority is most probably the originator of the object, and is determined by the system. In practice, principal authorities can be inferred based on other properties captured in SNO such as sn:owns or sn:created, that may be defined between an sn:User instance and an

Table 1: System-Level Access Control Policy Rules

Direct System Authorization:
$\mathcal{P} \wedge [?rsc \leftarrow \mathsf{sn}{:}p(s,o)] \Rightarrow \mathsf{ac{:}Permitted}(?rsc)$
Basic Authority Specification:
$\mathcal{P} \wedge [?rsc \leftarrow \mathsf{sn}{:}p(s,o)] \wedge \bigwedge_{i=1}^{n\geq 1} \mathsf{ac{:}uPermits}(u_i,\ ?rsc) \Rightarrow \mathsf{ac{:}Permitted}(?rsc)$
Conflict Resolution:
$\mathsf{ac{:}Permitted}(?rsc) \wedge \mathsf{noValue}(\mathsf{ac{:}Denied}(?rsc)) \Rightarrow \mathsf{ac{:}Granted}(?rsc)$

sn:Entity instance. We elaborate more on how this concept is used in our model in Section 3.2.2.2.

The access control policy rules in our model follow the syntax mentioned in Section 3.1.1. In order to efficiently express reified properties as protected resources in the rules we introduce the following shorthand.

Definition 1 (Reified Property Shorthand)**.** Expression $[?rsc \leftarrow \mathsf{sn}{:}p(s,o)]$ represents $?rsc$ as the protected reified property instance of type $sn : p$ that relates property subject s to property object o, i.e., $[?rsc \leftarrow \mathsf{sn}{:}p(s,o)] \equiv \mathsf{ac{:}RP}p(?x) \wedge \mathsf{ac{:}ropSbj}(?x,s) \wedge \mathsf{ac{:}ropObj}(?x,o)$.

In defining rule formats, we only use abstract authorization predicates. For instance, we use ac:uPermits in the format of personal authorizations. However, an actual rule needs to use one of its descendents as mentioned above. Moreover, we only provide syntax for positive authorizations for brevity reasons. The same syntax can be used to specify negative authorization (deny) rules.

3.2.2 System-Level Policy Rules

System-level access control policy rules are specified by SNS administrators. Table 1 shows the format of system-level policy rules. In these rules, $[?rsc \leftarrow \mathsf{sn}{:}p(s,o)]$ specifies the protected resource according to the reified property shorthand. $\mathcal{P}$ is a conjunction of zero or more of either SNO predicates, ac:Subject, or ac:hasPrincipalAuthority, and is used to characterize the target of the rule. We call $\mathcal{P}$ a *rule extension sentence*.

3.2.2.1 Direct System Authorization Direct permissions allow the system to authorize/deny accesses without involvement of user authorities. As shown in Table 1, a direct system authorization rule includes a rule extension sentence and a protected resource specification in the antecedent, and a ac:Permitted descendent as the consequence.

Example 1. The following two direct system authorization rules entitle everyone to read the properties of the objects for which they are principal authority.

$$\begin{aligned}&\mathsf{ac{:}Subject(?sbj)} \wedge \mathsf{ac{:}hasPrincipalAuthority}(?s,\ ?sbj)\\&\wedge [?rsc \leftarrow \mathsf{sn{:}property}(?s,?o)] \Rightarrow \mathsf{ac{:}PermittedRead}(?rsc).\\&\mathsf{ac{:}Subject(?sbj)} \wedge \mathsf{ac{:}hasPrincipalAuthority}(?o,\ ?sbj)\\&\wedge [?rsc \leftarrow \mathsf{sn{:}property}(?s,?o)] \Rightarrow \mathsf{ac{:}PermittedRead}(?rsc).\end{aligned}$$

In the first rule in the above example, the first two predicates in the body compose the rule extension sentence, and the third predicate specifies the protected property instance. The first predicate of the rule extension sentence indicates sbj to be the access subject, and the second one indicates sbj to be the principal authority for the protected property's subject. Note that ac:ReifiedProperty is the superclass of any reified SNO property. Therefore, the reified property shorthand applies to any property where variables s and o are bound to its subject and object. Finally, the consequence predicate allows read access to such a property for the access subject. The second rule is slightly different in that it considers the access subject to be the principal authority for the protected property's object.

3.2.2.2 Basic Authority Specification

Since most of the protected resources in an SNS are related to users' privacy, it is desirable that access control decisions are made based on the relevant users' policies rather than direct system authorization. In this respect, the role of system-level rules is to determine policy authorities for resources. According to Table 1, authority specification rules follow a similar format to direct system authorizations, i.e., having a rule extension sentence $\mathcal{P}$ and a reified resource specification in antecedent and an ac:Permitted descendent in consequence. But they also include a conjunction of ac:uPermits predicates, which indicates users whose authorization decisions matter regarding the characterized protected resources. This means that system authorizations are based on user-level authorizations.

Here, we propose a generic, basic authority model. Assuming there is a principal authority for every SNO class instance, as described in Section 3.2.1, it is safe to consider the same authority to be effective for any property associated with that class instance. Hence, the authority over an object property instance can be determined based on the principal authorities of the related class instances. For instance, access to sn:isFriendOf(Alice,Bob) is under the authority of both Alice and Bob. Analogously, the authority over a data property instance is the principal authority of the only related class instance. The basic authority model for read access (on an object property) can be expressed using the following rule.

$$\begin{aligned}&\textsf{ac:hasPrincipalAuthority}(?s,?u_1)]\wedge\ \textsf{ac:hasPrincipalAuthority}(?o,?u_2)]\\&\wedge\ [?rsc \leftarrow \textsf{sn:property}(?s,?o)]\\&\wedge\ \textsf{ac:uPermitsRead}(?u_1,\ ?rsc)\wedge\ \textsf{ac:uPermitsRead}(?u_2,\ ?rsc)\\&\Rightarrow\ \textsf{ac:PermittedRead}(?rsc)\end{aligned}\tag{3.1}$$

The above rule considers read access to a property authorized by the system only if both principal authorities of the class instances associated with the property authorize that access. Note that an SNS may specify and customize its own set of authority specification rules based on its application semantics. For instance, in many applications, it may be appropriate to allow delete operation over object properties with authorization of even one of the two associated principal authorities (e.g., deleting isFriendOf property between two users).

3.2.2.3 Conflict Resolution As mentioned in Section 3.2.1, OSNAC provides support for both positive and negative system authorization rules. The conflict resolution rule at the system level resolves any conflicts between such authorizations. We follow the *deny-overrides* conflict resolution policy. According to the definition of the rule in Table 1, the outcome of the rule is to grant access to the resource if the access has been authorized positively (i.e., being instance of ac:Permitted) and has not been authorized negatively (i.e., being instance of ac:Denied). Note that the rule does not follow SWRL syntax because of using the special operator noValue. The operator ensures not to grant the access if it is denied at the system level. We elaborate more on this operator in Section 3.2.4.1. A final grant decision is represented by assigning the resource to the appropriate descendent class of ac:Granted. Note that only class ac:Granted and its descendents represent the effective access control decision by the system. Other system-level and user-level authorization concepts represent merely intermediate decisions.

3.2.3 User-Level Policy Rules

User-level access control policy rules are specified by SNS users regarding the protected resources over which they have authority. The actual effectiveness of such rules is determined according to system authority policies. Table 2 shows the format of user-level policy rules. Similar to system-level policy rules, user-level policy rules include in antecedent a *rule extension sentence* $\mathcal{R}$, which is a conjunction of zero or more of either SNO predicates or ac:Subject, and a reified property shorthand for specifying a protected resource. Also, all user-level policy rules have an ac:uPermits descendent in their consequences. The first argument of this predicate has to be the user who specifies the rule. Otherwise, user authorities may be misused. We assume that SNSs enforce this requirement.

3.2.3.1 Personal Authorization A personal authorization rule expresses a permission granted by an individual user to others. According to the rule format shown in Table 2, a personal authorization rule uses a rule extension sentence and a reified property shorthand in antecedent and an ac:uPermits descendent in consequence.

Table 2: User-Level Access Control Policy Rules

Personal Authorization:
$\mathcal{R} \wedge [?rsc \leftarrow \mathsf{sn}{:}p(s,o)] \Rightarrow \mathsf{ac{:}uPermits}(u,?rsc)$
Dependent Authorization:
$\mathcal{R} \wedge [?rsc_1 \leftarrow \mathsf{sn}{:}p_1(s_1,o_1)] \wedge [?rsc_2 \leftarrow \mathsf{sn}{:}p_2(s_2,o_2)] \wedge \mathsf{ac{:}uPermits}(u,?rsc_1)$ $\Rightarrow \mathsf{ac{:}uPermits}(u,?rsc_2)$
Delegative Authorization:
$\mathcal{R} \wedge [?rsc \leftarrow \mathsf{sn}{:}p(s,o)] \wedge \mathsf{ac{:}uPermits}(u_2, ?rsc) \Rightarrow \mathsf{ac{:}uPermits}(u_1, ?rsc)$
Disjunctive Multi-Authority Specification:
$R = \{R_i\}$ $R_i = \mathcal{R}_i \wedge [?rsc \leftarrow \mathsf{sn}{:}p(s,o)] \wedge \mathsf{ac{:}uPermits}(u_i, ?rsc) \Rightarrow \mathsf{ac{:}uPermits}(pa, ?rsc)$
Conjunctive Multi-Authority Specification:
$\mathcal{R} \wedge [?rsc \leftarrow \mathsf{sn}{:}p(s,o)] \wedge \bigwedge_{i=1}^{n} \mathsf{ac{:}uPermits}(u_i, ?rsc) \Rightarrow \mathsf{ac{:}uPermits}(pa, ?rsc)$

Example 2. In the context of the running example (See Section 3.1.4), the following rule, expressed by Bob, authorizes his friends to read the photo-tags with which he has been marked.

$$\text{ac:Subject}(?sbj)\wedge \text{ sn:isFriendOf}(?sbj,\text{ bob})\wedge \text{ sn:PhotoUserTag}(?pTag)$$
$$\wedge\ [?rsc \leftarrow \text{sn:anonatesWith}(?pTag,\text{ bob})]\Rightarrow \text{ ac:uPermitsRead}(\text{bob},\ ?rsc)$$

In the above example, the first three antecedent predicates compose the rule extension sentence. The first and second predicates characterize the access subject to be Bob's friend, and the third predicate declares variable $pTag$ to be of type sn:PhotoUserTag. The forth predicate specifies the protected resource to be an sn:annotatesWith property that relates PhotoUserTag $pTag$ to Bob. The consequence predicate indicates that Bob authorizes such an access request.

3.2.3.2 Dependent Authorization Dependent authorization rules allow one authorization to be inferred based on another authorization. This is often useful in scenarios where authorizations need to be derived for related protected resources. As the format shown in Table 2, dependent authorization rules include a rule extension sentence, two protected property specifications, and an ac:uPermits descendent for the first protected property in antecedent, and an ac:uPermits descendent for the second protected property in consequence. In other words, dependent authorization allows for authorization propagation from protected resource rsc_1 to protected resource rsc_2 under specific condition determined by rule extension sentence $\mathcal{R}$.

Example 3. In the context of the running example (See Section 3.1.4), suppose Alice has specified detailed authorization rules for different types of annotations on her photo. Then she can make sure whoever gets to access the annotations can also access the photo content using the following rule, without redefining all the restrictions.

$$[?rsc_1 \leftarrow \text{sn:annotates}(?x,\text{ photo1})]\wedge\ [?rsc_2 \leftarrow \text{sn:hasContent}(\text{photo1},\ ?c)]$$
$$\wedge\ \text{ac:uPermitsRead}(\text{alice},\ ?rsc_1)\Rightarrow \text{ ac:uPermitsRead}(\text{alice},\ ?rsc_2)$$

3.2.3.3 Delegative Authorization Delegation has been shown to be useful in the context of access control models [6]. We observe that delegating authority improves the flexibility of policies in OSNAC. Based on a delegative authorization, a user delegates its authority over a specific resource to another user. According to the rule format shown in Table 2 user u_1 delegates authorization on a specific resource to user u_2. In other words, user u_1 respects the authorizations made by user u_2 on that resource. Delagative authorizations may be used to relax authority on the protected properties. The basic authority specification rule stated in Section 3.2.2.2 (rule (3.1)) requires both principal authorities of the associated class instances of a property to authorize a read access, in order for it to be authorized by the system. However, such a mutual agreement might be too restrictive in some contexts. The two end authorities can use delegative authorizations to respect one another's decisions on the specific access of choice, without a change in system-level rules.

Delegative authorizations are very flexible and secure in terms of delegation power. First, an authority can flexibly customize the permission. For instance, it can restrict the target subjects to have certain characteristics, or the resource/action to be of certain type. Second, subsequent updates to the delegative authorization rule will be applied seamlessly, without a need to worry about grant/revoke propagation issues that delegation models usually deal with. This is because unlike traditional delegation models, the authorizations are not explicitly transferred; the authorization rule is the sole means of delegation. Third, since delagative authorizations are at the user level, there is no need to assure that the delegator actually has the authority on the resource. Only the valid delegations will be effective based on the system-level authority specification rules.

3.2.3.4 Multi-Authority Specification There are scenarios in SNSs where it is desirable to have more authorities to weigh in an access control decision than just the directly related authorities. We support multi-authority specification in two ways. A principal authority may use multiple delegative authorization rules to enable a *disjunctive multi-authority*. Such a multi-authority is disjunctive in the sense that an authorization by any corresponding authority in the set is a sufficient condition for that permission to be considered authorized by the principal authority. Alternatively, a principal authority may create

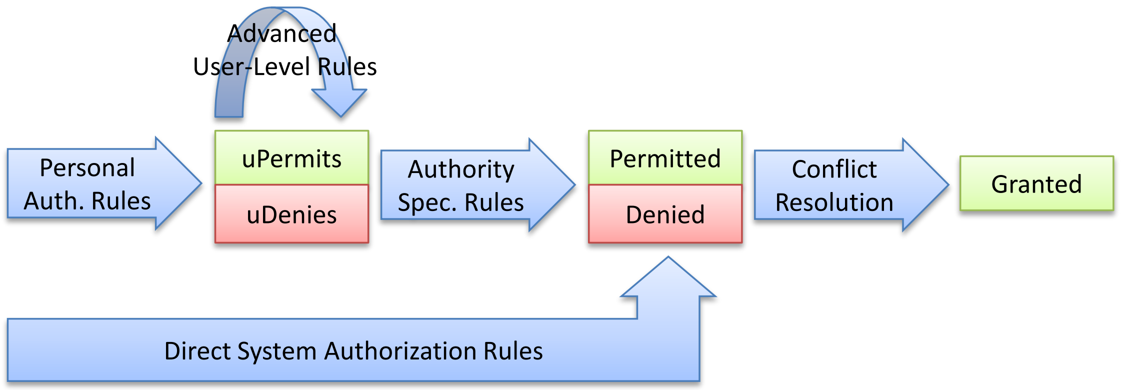


Figure 8: OSNAC Access Control Policy Rules Entailment

a *conjunctive multi-authority*, in which every involved authority is required to authorize an access in order that it would be considered authorized by the principal authority. Table 2 shows the formats of a rule set and a single rule that establish disjunctive and conjunctive multi-authority, respectively, where principal authority user pa shares the authority with users u_1, u_2, …, and u_n.

3.2.4 Access Control Enforcement

3.2.4.1 Entailments and Conflict Resolution We presented the syntax of OSNAC policy rules in Sections 3.2.2 and 3.2.3. Figure 8 demonstrates the overall entailment process based on such rules. The blue arrows represent policy rules showing how they entail new policy concepts/relationships. Properties ac:uPermits and ac:uDenies represent access control decisions at the user level. They are entailed based on either ontology (using personal authorization rules) or other already entailed access properties (using advanced user-level rules such as dependent or delegative authorizations). Classes ac:Permitted and ac:Denied represent initial access control decisions at the system level. Class ac:Granted represents the final access decision. Entailment of this class for the protected resource means granting the access, and denial otherwise.

The green and red colors in Figure 8 represent positive and negative authorization concepts, respectively. Both user-level and system-level rules allow for positive and negative authorizations. Therefore, an access might be both positively and negatively authorized at each level. The conflict resolution is performed based on dedicated conflict resolution rule as the last step of entailments (refer to Section 3.2.2.3). The conflict resolution will either entail ac:Granted for the protected resource or not, equivalent to final grant or deny decision, respectively.

As mentioned in Section 3.2.2.3, we leverage a built-in operator in Jena rules, i.e., noValue. Therefore, the conflict resolution rule is not pure SWRL. This operator simulates a weak notion of negation. SWRL does not support negation-as-failure due to the open world assumption of the Semantic Web: if you cannot entail a predicate, you cannot entail that it does not exist. In the case of OSNAC, that means if a negative authorization cannot be entailed, we are not sure if it exists or not. However, from an access control policy perspective, the condition is satisfactory. Operator noValue can exactly perform that for us. According to the conflict resolution rule, if the rule engine entails Permitted(?rsc) it will next try to entail the negative authorization (Denied(?rsc)). If such a reasoning is unsuccessful, noValue(Denied(?rsc)) will be true, and the access will be granted.

In addition to the final conflict resolution, there is possibility of partial conflict resolution at intermediate stages. The basic authority specification rules at the system level (See Section 3.2.2.2) aggregate user-level authorizations. Therefore, they can partially resolve some conflicts. For instance, suppose a basic authority specification rule considers positive authorization of one of the two users in a friendship relationship satisfactory to delete that relationship. Such a rule indirectly resolves conflicts between a positive user authorization and a negative user authorization, in case it exists.

3.2.4.2 Access Authorization

A basic access request is a triple $\langle sbj, rsc, act\rangle$, where sbj is the user who requests the access (instance of sn:User), $rsc = p(s, o)$ refers to the property instance to be accessed (instance of ac:ReifiedProperty), and act is the action requested (read, delete, or insert). Making access deciding in OSNAC involves reasoning on SNS ontology using access control policy rules. The ontology and policy rules are expressed using OWL-DL and DL-safe SWRL rules, respectively. The following definition captures the notion of access authorization.

Definition 2 (Access Authorization)**.** Given an access request $\langle sbj, rsc, act \rangle$, the access is granted if and only if, given the fact ac:Subject(sbj), predicate ac:Granted(rsc) can be entailed (GrantedRead, GrantedDelete, or GrantedInsert, depending on act). The access is denied otherwise.

The above access authorization process is expected to determine a correct decision for any access request.

Theorem 1. *Access authorization in OSNAC is correctly enforced.*

Proof. Access authorization in OSNAC involves trying to entail ac:Granted by resolving any conflicts between ac:Permitted and ac:Denied entailments at the system level. According to Motif et al.'s work [60], SWRL entailments are decidable if the language is limited to OWL-DL and DL-safe SWRL rules. This applies to our OSNAC ontology, and user-level and system-level policy rules (except conflict resolution). Therefore, entailing ac:Permitted/ ac:Denied is decidable. As discussed in Section 3.2.4.1, the conflict resolution in the last stage of authorization is a straightforward step once the ac:Permitted/ac:Denied entailments are known. It involves testing a condition, which is clearly decidable. Therefore, on one hand, entailment of ac:Granted and access authorization is complete in OSNAC, meaning that for every access request there will be an access decision entailment. On the other hand, since OSNAC is based on Semantic Web technologies which themselves are based on Description Logics, such entailments are sound. We conclude that access authorization entailment in OSNAC is both sound and complete and therefore is correctly enforced. □

3.2.4.3 Query Augmentation

We note that in the case of information retrieval from an SNS knowledge base multiple relations may be queried and evaluated simultaneously in order to retrieve a result set of interest. Conceptually, for each valid variable assignment in a query, every bound relation needs to be considered as one basic access authorization. However, checking access authorization per such relation is not efficient, and needs modification of the retrieval engine. Alternatively, we augment a query with access check primitives and evaluate the augmented query in order to retrieve only the authorized results.

Theorem 2 (Query Access Authorization). *Let pair $\langle sbj, Q\rangle$ be a query Q on SNO submitted by subject sbj. Let $Q_W = \bigwedge_{i=1}^{n} \textsf{sn:}p_i(s_i,o_i)$ represent the conjunctive* WHERE *clause of query Q. Given the fact $\textsf{ac:Subject}(sbj)$, a retrieval engine can automatically enforce access authorization and retrieve the authorized result by evaluating:*

$$Q'_W = \bigwedge_{i=1}^{n} \{\textsf{sn:}p_i(s_i,o_i) \wedge [?rsc_i \leftarrow \textsf{sn:}p_i(s_i,o_i)] \wedge \textsf{ac:GrantedRead}(?rsc_i)\}$$

In the above theorem, each relation predicate in the original query is followed by two predicates for access control purpose: the first predicate bounds the relation to a resource variable, and the second predicate checks if the subject is authorized to access the resource. A query that is augmented with access primitives can be directly processed by a query retrieval engine on the ontology. The access control policy rules are enforced seamlessly using a reasoner without the need for modifying the underlying retrieval engine.

Example 4. Suppose Bob requests access to the list of Alice's friends who reside in Pittsburgh. This is a complex query that involves accessing the list of Alice's friends, where they live, and their names. The following is a SPARQL-like syntax for this query.

```
SELECT ?x ?fullname
WHERE{ sn:isFriendOf(alice, ?x)
       ∧ sn:residesIn(?x, pittsburgh)
       ∧ sn:hasFullname(?x, ?fullname) }
```

Its access-augmented *WHERE* clause will be as follows.

```
sn:isFriendOf(alice,?x) ∧
∧ [?rsc1 ← sn:isFriendOf(alice, ?x)]∧ ac:GrantedRead(?rsc1)
∧ sn:residesIn(?x, pittsburgh)
∧ [?rsc2 ← sn:residesIn(?x, pittsburgh)]∧ ac:GrantedRead(?rsc2)
∧ sn:hasFullname(?x, ?fname)
∧ [?rsc3 ← hasFullname(?x, ?fname)]∧ ac:GrantedRead(?rsc3)
```

By executing the augmented query in Example 4, if Bob does not have access to even one of the relations in the query corresponding to a specific Alice's friend, that person's information will not be retrieved. Thus, the result set reflects the authorized information according to the access control policies.

3.2.5 Evaluation

In this section, we evaluate our proposed access control approach in terms of expressiveness and execution performance. We enumerate a set of access control policy features that are useful in SNS scenarios, and compare our approach against the literature in achieving them. We also present a prototype implementation of OSNAC and report the performance results of our prototype.

3.2.5.1 Expressiveness We evaluate the expressiveness of OSNAC from two perspectives. First, we show that our ontology-based data model is rich enough to capture useful specification requirements for protected resource that have been proposed in the literature which may be also useful for privacy control in SNSs. Second, we enumerate a set of access control policy features that are useful in SNS scenarios, and compare our approach against the literature in achieving them. We consider models in the literature that are either targeted for SNSs or are generic and feature-rich enough that may be useful in these environments.

We specifically compare our approach with relationship-based policies [14, 15, 23, 3, 24, 25], Carminati et al.'s Semantic Web-based policy framework [13], access control for RDF stores [66], and XACML [2].

Based on our literature review in Section 2.1, we consider the following features in characterizing protected resource in access control policies in the literature.

- *Resource identity*: The basic approach is to use a resource's identity in access control policies. The relationship-based access control approaches for SNSs (See Section 2.1.1) follow this approach as they are only concerned about characterizing access subjects using relationships. In our data model, this is equal to protecting a class instance (e.g., a specific photo) as a whole. OSNAC can support this by specifying any property that is related to that instance as protected resources. Note that the specification can be abstract, i.e., there is no need to enumerate all possible properties for an instance.

- *Resource attributes*: A more flexible way is to use resource attributes in the policies. XACML, a widely popular access control standard, supports attributed-based specification of the elements of access control policies including protected resources. In our ontology-based approach, properties of a class instance will act as its attributes. A typical attribute-value pair in XACML can be represented using a data property and its data value. Moreover, object properties can be leveraged to specify relationship to other object instances as attributes. Note that, here, the attributes are only used to identify the protected resource, unlike in our model that can consider attributes as protected resources.

- *Resource hierarchies*: Many access control models support hierarchical organization of protected resources [70, 18], where access control rules may be specified at different levels. Usually, access control policies are propagated downwards in the hierarchy. Our data model supports hierarchies of both classes and properties. Instances of a class are considered instances of the ancestor classes too. This ensures that a policy applicable to a class of instances will be also applicable to instances of any descendent classes.

In the above, we showed that OSNAC data model at the least supports conventional protected resource specification approaches in the related literature. In the followings, we enumerate a set of useful features for policies in SNSs that are supported in OSNAC, and compare it to the proposed models in the literature. We draw sample access scenarios from our running example (See Section 3.1.4) to elaborate on these comparisons.

- *Networked data*: As we previously discussed, an ontology-based data model is a good fit for SNSs to represent its inherent networked data: i.e., user to user, object to object, and user to object relationships. For example, suppose Alice wants to specify a policy regarding access to the content of photos in which her close friends are tagged. This characterization cannot be specified using a simple attribute. The closest approach in the literature to capture this is policies based on resource content in XACML. However, in order to achieve this, one should consider the whole photo, its annotations, and even friendship relationships as one protected resource in an XML document. The problem is that the protected resource becomes very coarse grained. In contrast, one can easily express such a policy in the context of OSNAC, using network semantics.

- *Objects vs. semantics of objects*: Access control models in the literature consider whole objects as protected resources. Objects are translated to class instances in the ontology context. OSNAC provides finer-grained specification of protected resources by considering selective aspects of the semantics of objects. In an ontology, semantics of class instances are modeled using properties, which are recognized as protected resources in OSNAC. For instance, the friendship between Bob and Carol is part of the semantics of their representative object instances, and therefore can be considered as a protected resource. Considering ontology properties as protected resources makes it flexible to specify policies independently for each aspect of the semantics. For instance, one may be able to access Bob's tag in photo1 without accessing the "who has created the tag" information. Or one can see the contents of photo1 without accessing the "who owns this photo" or "who likes this photo" information.
- *Hierarchies*: As mentioned earlier, our data model supports class and property hierarchies. This enables propagation of policies down the hierarchies, allowing for flexibility of specifying policies at different levels of abstraction. For instance, Bob can specify a policy regarding properties relating him to UserTag instances, and such a policy would automatically apply to both PhotoUserTag and TextUserTag instances. As another example, a policy regarding isFriendProperty will be also applicable to isCloseFriendProperty.
- *Authority of SNS and users*: Access control models have been mainly used to provide security in an organizational context. Therefore, the organization or system, has been considered the stakeholder and authority of specifying policies. In contrast, in the context of SNSs, users should be considered stakeholders too as they should be able to manage the privacy of their related information. Realizing the need for both system and user authorities, OSNAC supports administrating this policy system. We considered the practicality of our approach for real-world products by recognizing system's authority as the final effective decision maker in this environment. To the best of our knowledge, our work is the first approach to tackle this problem.

- *Multiple authorities*: A challenging aspect of policy specification in SNSs is the existence of multiple stakeholders that have interest in specifying policies for the same protected resources. Most of the proposed access control models for SNSs, such as relationship-based models (See Section 2.1), do not provide support for multiple authorities. OSNAC recognizes multiple authorities in two ways. We consider a principal authority per class instance. The authority for properties (i.e., protected resources) is determined by the principal authorities of the associated class instances. This means that for every object property there will be at most two principal authorities involved. A potential authorization conflict between the involved authorities can be resolved either based on either the basic authority specification rules or the conflict resolution rule at system level (See Section 3.2.2). The second way of supporting multiple authorities in OSNAC is through use of advanced user-level policies (multi-authority rules). In that case, the potential conflicts are resolved based on the multi-authority rules themselves (See Section 3.2.3.4). We note that there have been few recent work aiming specifically to tackle the problem of policy authoring by multiple users [72, 73, 38]. OSNAC provides the high-level language using which the result of such specific policy authoring approaches can be specified.

- *Negative authorizations and conflict resolution*: Supporting negative authorizations (in addition to positive authorizations) enables more flexible policy specification and management [7, 70, 74]. However, co-existence of positive and negative authorizations may lead to conflicts, which would need to be resolved to make an access decision. In OSNAC, we support negative authorization specification both at the user and system level. As explained in Section 3.2.4.1, we resolve conflicts in the last inference step based on system-level authorizations. We chose to use the *deny-overrides* strategy for this purpose, as a simple and flexible approach. Using this strategy in a closed system assumption lets the users (or SNS) grant access using positive authorizations and make exceptions using negative authorizations. The related access control models for SNSs (i.e., relationship-based policies and Carminati et al.'s Semantic Web-based policy framework) do not support negative authorization. There are other conflict resolution approaches in the access control literature [59, 55, 21] such as *more-specific-overrides*, which is more suitable for systems with hierarchical organization of protected resources. Also, XACML supports conflict resolution in the form of rule and policy combining algorithms. The combining algorithms define procedures to determine access decision based on the evaluation of multiple rules/policies.

We summarize the comparison factors discussed above in Table 3.

Table 3: OSNAC Expressiveness Comparison (❋ indicates partial support)

Feature \ Model	OSNAC	Relationship-Based [14, 15, 23, 3, 24, 25]	Semantic Web-Based [13]	Access Control for RDF [66]	XACML [2]
Networked Data	✓	❋(user-user)	✓	❋(object-object)	✗
Controlling Semantics	✓	✗	✗	✓	✗
Hierarchies	✓	✗	✓	✓	✗
Policies for SNS/Users	both	user only	user only	system only	system only
Multiple Authorities	✓	✗	✗	✗	✗
Negative Authorization	✓(deny-overrides)	✗	✗	✗	✓(flexible algorithms)

3.2.5.2 Prototype Implementation We have developed a prototype access control engine based on our proposed OSNAC model to protect an SNS knowledge base. Figure 9 illustrates the architecture of the prototype implementation. Access control policy rules are provided by users and system administrators, using separate interfaces, and are stored in the policy rule-base. Rules are expressed using SWRL as explained in Section 3.2. However, since SWRL is not directly supported in Jena, we programmatically convert rules to Jena's own rule language in a policy compilation phase. Note that there is no loss of expressiveness in this process. At run time, the *User Request Processor* accepts the requests from a user (in fact, from the SNS on behalf of a user), and passes it to the *Query Modifier* module, where it is augmented with access control primitives (refer to Section 3.2.4). The modified query is then sent to the *SPARQL Engine*. Before execution of the query by engine, a fact is inserted in the knowledge base asserting the user to be the access subject. The SPARQL Engine then interacts with the SNO and ACO to retrieve the query results. In the retrieval process, the *Access Inference Engine* employs Jena general purpose rule engine to infer access primitive predicates (i.e., ac:authorizes and ac:Permitted) based on the knowledge stored in the ontologies and according to the access control policy rules. The access subject assertion is removed from knowledge base after query has been executed. Finally, the authorized query results are returned to the *User Request Processor*.

We implemented the prototype in Java based on the Apache Jena Semantic Web framework[1]. The knowledge base (social networking system ontology) and access control ontology are expressed in OWL-DL. We use Jena's general purpose rule engine for reasoning about accesses. For this purpose, we convert the access control policy rules from SWRL-DL into the Jena's rule syntax (which is very similar to SWRL-DL). We use the backward chaining strategy of the rule engine for reasoning. Th Jena's reasoning power has a limitation in that it is an in-memory rule engine. Therefore, we cannot reason about large ontologies that one may deal with in a real production system. But it is suitable for our small prototype implementation and performance evaluation purpose.

[1]http://jena.apache.org/

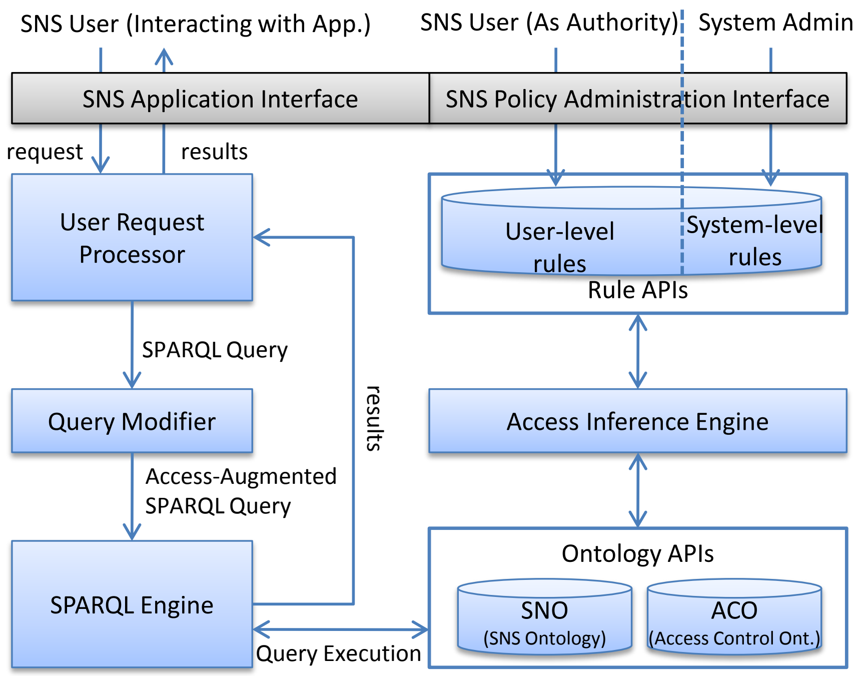


Figure 9: Architecture of the Prototype Implementation

3.2.5.3 Experimental Results We manually tested the policy enforcement by the prototype OSNAC engine, submitting SPARQL query requests for a knowledge base that captures the running example (See Section 3.1.4). As for system-level policies, we use the basic authority specification described in Section 3.2.2.2. Moreover, two direct system authorizations allow each user's full access to the properties that involve her or her owned digital objects. As for user-level policies, we consider two personal authorization rules for each user that allows her friends to see the properties involving her or her objects. Additionally, a delegative authorization rule by Alice delegates authorizations on properties of photo1 to Bob, and a personal authorization rule by Bob allows read access to them by his friends. The engine successfully returns only the expected authorized information. For example, David is granted a read access to hasContent(photo1, photo1_data), as a result of Bob's authorization which itself is possible because of Alice's delegation.

In order to evaluate the feasibility of employing our engine, we assessed its time performance in resolving access decisions using synthetic knowledge bases. For this purpose, we decided to focus on a scenario where users want to control their friendship links. In such a scenario, we are able to evaluate the effects of change in the number of users and their protected resources (i.e., friendship links) on access control performance, without the need to consider multiple complicating variables. We generate two sets of synthetic datasets, to individually evaluate the effect of each factor. First, in order to evaluate the effect of the number of users, we generate four datasets with increasing number of users (i.e., 1250, 2500, 5000, 10000 users). In each dataset, users have friendship links based on a random network, generated using Erdos-Renyi model, with the fixed link probability of 0.2. Second, in order to evaluate the effect of network density and number of protected resources, we generate four datasets with fixed number of users (i.e., 5000) and increasing number of friendship links (i.e., link probabilities 0.1, 0.2, 0.3, 0.4, and 0.5). At the system level, we consider the basic authority specification rules specified in Section 3.2.2.2 and direct authorization of properties related directly to a user to herself. At user level, we consider the same policy for all users, where they allow their friends to access their friendship relationships. We run the prototype on a standard desktop PC. We measure *access check time*, i.e., the time it takes for the engine to decide about an access authorization query. We report the average access check time for 1000 random access authorizations in each of which a randomly chosen user requests access to a randomly chosen friendship link. The engine takes few seconds to load the ontologies and inference model. Since such initialization is performed only once, we only present our evaluation of access check time. Table 4 summarizes the above-mentioned experimental setup for our feasibility test.

Figures 10 shows an acceptable performance by our prototype engine when the user population is increased. For instance, it takes about 6 milliseconds on average to check access decision for a random network of 5000 users with friendship probability of 0.2. As expected, the access check time increases with the increase in the number of users, with a semi-linear trend. As the user size increases, in addition to larger dataset, there will be more access control policy rules that needs to be processed by the reasoner, which leads to slower authorization decision.

Figure 11 depicts that access check time has a semi-linear growth by increase in the network density. By increasing the probability of friendship links, processing the personal authorization rules in our policies requires verifying if the access subject is among a larger number of users who are friends with the two sides of the protected friendship relationship. Therefore, the engine needs to spend more time to reason about authorization.

Table 4: Experimental Setup for Feasibility Test

	Experiment 1	Experiment 2
Knowledge Base	Erdos-Renyi random network, #nodes: 1250/2500/5000/10000, Friendship probability: 0.2	Erdos-Renyi random network, #nodes: 5000, Friendship probability: 0.1/0.2/0.3/0.4/0.5
System-Level Policies	Basic authority specification (Rules (3.1)) Two direct system authorization rules for self access: ac:Subject($?u$) $\wedge$ ac:hasPrincipalAuthority($?s$,$?u$) $\wedge$ [$?rsc \leftarrow$ sn:property($?s$,$?o$)] $\Rightarrow$ ac:PermittedRead($?rsc$) ac:Subject($?u$) $\wedge$ ac:hasPrincipalAuthority($?o$,$?u$) $\wedge$ [$?rsc \leftarrow$ sn:property($?s$,$?o$)] $\Rightarrow$ ac:PermittedRead($?rsc$)	
User-Level Policies	For each user u_i, two personal authorization rules: ac:Subject($?y$) $\wedge$ sn:isFriendOf(u_i,$?y$) $\wedge$ [$?rsc \leftarrow$ sn:isFriendOf(u_i,$?x$)] $\Rightarrow$ ac:uPermitsRead($?rsc$) ac:Subject($?y$) $\wedge$ sn:isFriendOf(u_i,$?y$) $\wedge$ [$?rsc \leftarrow$ sn:isFriendOf($?x$,u_i)] $\Rightarrow$ ac:uPermitsRead($?rsc$)	
Metric	Average access check time for 1000 requests each by a randomly selected user accessing a randomly selected friendship link	

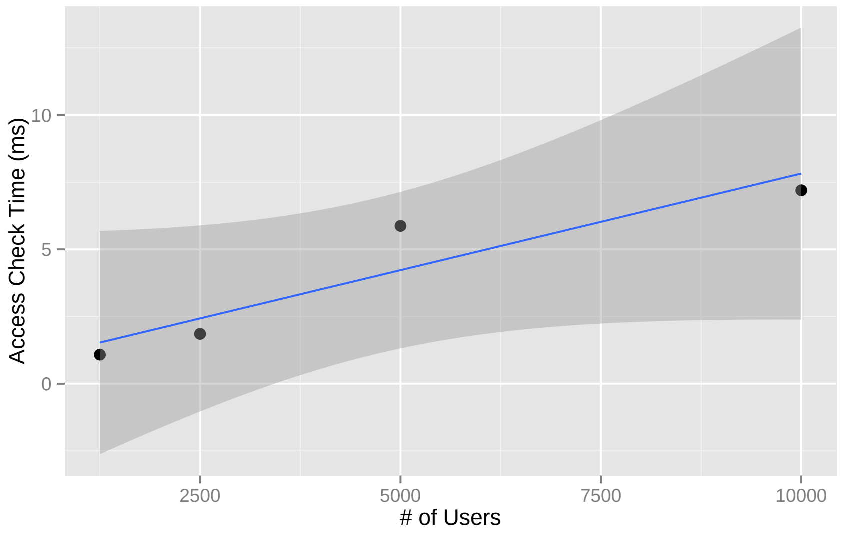


Figure 10: Prototype Performance for Different Number of Users (Friendship Probability = 0.2)

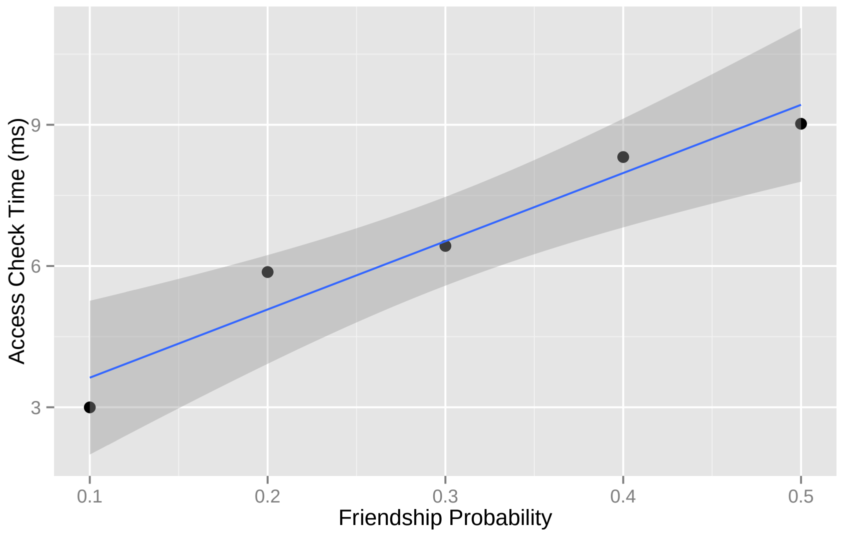


Figure 11: Prototype Performance for Different Network Densities (5000 Users)

3.3 PRIVACY CONTROL POLICY ANALYSIS FRAMEWORK

In this section, we present our policy analysis framework for SNSs based on ontology-based privacy control policies. We lay out the foundations of ontology-based policy analysis and provide details on how various policy properties can be analyzed by leveraging such a framework. The framework shares the same foundation of modeling protected resources with OSNAC (See Section 3.1), however, it is different in the way it models policies.

3.3.1 Representing Policies

For the purpose of policy analysis framework, we are interested in the following policy characteristics:

- *Fine-grained specification*: Ontology-based specification of knowledge in SNS and use of reification techniques, as described in Section 3.1, allows us to provides a fine-grained method to specify permissions and access subjects.
- *User/System authors*: We require that policies can be specified about protected resources by both users and system as policy authors.
- *Positive/Negative authorization*: Finally, policies should support both positive and negative authorization in order to allow a flexible specification.

Given the above requirements, we consider five components to policy specification: *protected resource, action, subject, effect (grant/deny), and author*. While these components allow very expressive policy specification, we follow a simple approach to combining policies for the purpose of the policy analysis framework. In the analysis framework, we consider manual combination of policies through analysis operations as it will be explained in Section 3.3.2.2. For example, a deny-overrides conflict resolution meta-policy can be achieved using subtraction operation. However, while various flexible combining operations are supported in our framework we do not discuss meta-policies that require algorithmic procedures in our policy analysis framework. The reason behind this limitation is to constrain the framework within the expression power of OWL-DL language for decidability purposes. Note that it means that our policy analysis framework will not leverage SWRL rules unlike OSNAC.

The core concept used for policies in our analysis framework is class Access, which characterizes an access request. The following properties characterize an Access instance:

- pResource: specifies the protected resource (i.e., reified property) that is accessed. The domain and range of object property pResource are Access and ReifiedProperty, respectively.
- pAction: specifies the action on the protected resource. We consider basic data manipulation actions of selection, insertion, and deletion of protected resources in the knowledge base. The domain and range of object property pAction are Access and PolicyAction, respectively. Class PolicyAction is comprised of individuals pa_select, pa_insert, and pa_-delete).
- pSubject: specifies the subject who accesses the protected resource. The domain and range of object property pSubject are Access and User, respectively.

As an example, suppose David wants to know if Bob and Carol are friends with each other. The corresponding Access instance can be characterized by the following predicates, where rop_isFriendOf_bob_carol is the reified property representing friendship between Bob and Carol:

access1 pResource rop_isFriendOf_bob_carol

access1 pAction pa_select

access1 pSubject david

A user policy is expressed by specifying a set of applicable accesses and indicating the user's intended effect for such accesses, i.e., either grant or denial. In our policy analysis framework, such an effect is specified using properties permits or denies, respectively. Both these properties have domain of User and range of Access. Formally, a user policy is defined as a concept that characterizes a subset of class Access and is also defined as a subset of intended authorization relationship. For example, the following two predicates specify Bob's policy that allows his friends to access his friendship relationships.

$$\begin{aligned}
\mathsf{up1} &\equiv \mathsf{Access} \\
&\quad \mathsf{and\ pResource\ some\ (RPisFriendOf\ and} \\
&\quad \mathsf{(ropSbj\ value\ bob)\ )} \\
&\quad \mathsf{and\ pAction\ value\ pa\_select} \\
&\quad \mathsf{and\ pSubject\ some\ (User\ and\ (isFriendOf\ value\ bob))} \\
\mathsf{up1} &\sqsubseteq \mathsf{inverse\ permits\ value\ bob}
\end{aligned}$$

The first predicate above defines user policy concept up1 where the protected resource is a friendship reified property with Bob as its subject, action is selection, and the subject is a friend of Bob. The second predicate indicates that such accesses are among those authorized by Bob. Note that the inverse of property permits is used in the second predicate to switch the place of its domain and range.

3.3.2 Analysis Tasks

We define a privacy analysis task as the act of performing *analysis operations* on the *analysis target* based on *analysis factors*.

3.3.2.1 Analysis Targets Analysis targets are characterizations of accesses or their related components, i.e., protected resources, actions, and subjects. Such characterizations are specified using class Access and its related properties, that were described in Section 3.3.1. While a full characterization of access uses all three properties (pResource, pAction, and pSubject), a partial characterization of access uses a subset of those properties. For instance, using only properties pResource and pAction, one can characterize permissions and analyze them in our framework.

3.3.2.2 Analysis Operations Analysis operations either combine or reason about two analysis targets. In order to avoid unnecessary complication, we provide definition of the operations based on full characterization of accesses. However, the same operations are applicable to partial characterization of accesses. The only requirement is that the operands of analysis operations should be of the same type.

The basic operations that allow analysis in our framework are notions of complement and intersection of concepts as well as subsumption reasoning. Let concepts P and Q be subclasses of Access. The following list include the two primary combining operations and the primary reasoning operation in our framework:

- not P (Negation): denotes whatever that is not described by P.
- P and Q (Intersection): denotes accesses that are described by both P and Q.
- P $\sqsubseteq$ Q (Subsumption): if satisfied it means that all accesses described by P are also described by Q.

The above operations can be conveniently supported by an OWL reasoner. We derive other operations and reasoning by combining the primary operations. Some of the typical derived operations include:

- P or Q (Union): denotes accesses that are described by either P or Q (or both). Such a concept is equivalent to not((not Q) and (not Q)).
- P $\setminus$ Q (Subtraction): denotes accesses that are described by P and not by Q. Such a concept is equivalent to P and not Q.
- P $\equiv$ Q (Equivalency): if satisfied it means P and Q describe the same set of accesses. Such a concept is equivalent to satisfying both P $\sqsubseteq$ Q and Q $\sqsubseteq$ P.

Suppose we have two analysis targets T_1 and T_2, where T_1 specifies *read accesses to Bobs friendship relationships by his friends* and T_2 specifies *read accesses to any of Bobs relationships by his close friends*. The interpretation of applying each of the above-mentioned analysis operations on these targets are listed in Table 5.

3.3.2.3 Analysis Factors We consider several *analysis factors* based on which the analysis operations can be performed on the analysis target. Two classes of analysis targets can

Table 5: Interpretation of Analysis Operations on Sample Targets

Operation	Interpretation
not T_1	Any access except reading Bobs friendship relationships by his friends
T_1 and T_2	Read access to Bobs friendship relationships by his close friends
$\mathsf{T}_1 \sqsubseteq \mathsf{T}_2$	Are read accesses to Bob's friendships by his friends subsumed by those to any of his relationships by his close friends?
T_1 or T_2	Read accesses to Bob's friendships by his friends or to any of his relationships by his close friends
$\mathsf{T}_1 \setminus \mathsf{T}_2$	Read accesses to Bob's friendships by his friends who are not his close friends
$\mathsf{T}_1 \equiv \mathsf{T}_2$	Is the set of read accesses to Bob's friendships by his friends equivalent to the set of read accesses to his relationships by his close friends?

be combined or compared according to their different respective analysis factors:

- *Policy Effect*: User policies state either of the two possible effects: grant or denial. Those are also known as positive and negative effects, respectively. For example, a subtraction operation between positive and negative policies allows inferring authorized accesses after resolving conflicts.
- *Policy Author*: The policy author is the entity that provides the policy. According to our policy representation, the policy author is specified as the subject of permits/denies predicates. For example, one can compare and contrast two users' policies with regards to their same type of protected resources (e.g., their photos).
- *Time State*: We can consider different time states (snapshots) for an SNS. By considering the same policy in different time state of a system, one can perform time-based analysis tasks such as analyzing evolution of policies.
- *Logical State*: Logical states express classes of analysis target that do not exist as part of the current policies in an SNS. For example, one can compare what should ideally be the policies in an SNS versus what actually exists in the system. As another example, we

can constrain the scope of the analysis target by calculating the intersection of a limiting logical state and the analysis target at hand.

- *System*: Ontology-based model of the data and policies allows us to map between the concepts in one SNS and the respective ones in another SNS. Therefore, system itself can act as analysis factor. For example, we can verify the equivalence of the same user's policies in two different SNSs. As another example, combination operations can provide a global view of the policies that a user has in several SNSs.
- *Access Components*: Those components of access that are not part of the analysis target could also be employed as analysis factor. For example, if the target of analysis is permission, we can compare them according to different set of access subjects.

Figure 12 shows a conceptual visualization of an analysis task. We represent access characterization on a two-dimensional plane, where the first dimension represents protected resources and actions and the second dimension represents access subjects. Note that a more accurate representation was to use separate dimensions for each component of access characterization. But we avoid it to represent the analysis factors on the third dimension. In the figure, T_1 and T_2 are two access characterizations that differ on their corresponding analysis factor value. The analysis task performs one of the operations described in Section 3.3.2.2 on T_1 and T_2 based on their differences according to the analysis factor. For example, if T_1 specifies accesses that Alice authorizes and T_2 specifies accesses that Bob authorizes, T_1 and T_2 will be corresponding to accesses that both Alice and Bob authorize, based on the policy author analysis factor. As another example, if T_1 specifies accesses that a user allows and T_2 specifies accesses that the same user denies, $\mathsf{T}_1 \setminus \mathsf{T}_2$ will be corresponding to accesses that are authorized by the user after conflict resolution, based on the policy effect factor. More complex analysis tasks can be formed by performing a series of such basic analysis tasks.

Table 6 lists a number of sample analysis tasks that can be carried out using our framework, showing their corresponding analysis targets and factors. A star symbol for an analysis factor in the table indicates that it can be involved in the analysis task depending on the purpose of analysis. For example, the policy effect analysis factor has been marked as optional for many of the proposed tasks in the table. In that case, depending on the policy model and

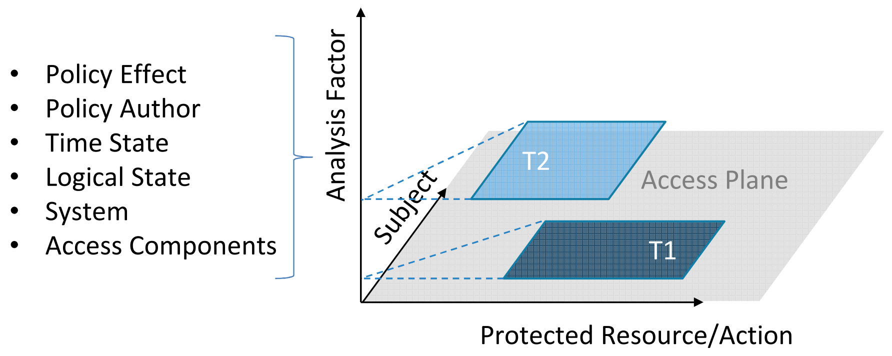


Figure 12: Privacy Control Policy Analysis Task

the task at hand, if policies with both positive and negative effects exist a combining step may be required to resolve conflicts (subtracting the targets in the negative policies from their corresponding targets in the positive policies).

3.3.3 A Detailed Analysis Task: Completeness of Privacy Control Permissions

In this section, we provide details for Task 1 in Table 6. In a typical SNS such as Facebook, there exists different privacy settings that can be configured by a user in order to control others' access to the information related to her. These settings are in fact equivalent to access control policies that are expressed by the users for the respective digital objects. Such privacy settings are by no means complete in the sense that they do not control access to all the potentially privacy-sensitive information about a user. That is the case even for Facebook, which has fairly the most extensive set of privacy settings among SNSs. Access to the rest of the information related to the user is governed by a set of fixed, default rules set by the SNS itself.

There is a major issue with the current practice of privacy control policies in SNSs such as Facebook. The current privacy settings do not provide users with adequate power to protect their privacy-sensitive information. Moreover, the default policies enforced by an SNS are not clearly described to the users. Therefore, users are unsure about what to expect from the system. Users need to learn about them either by harvesting help pages in the SNS

Table 6: Sample Analysis Tasks (❋ indicates optional analysis factor)

		Analysis Target			Analysis Factor							
	Analysis Task	Subject	Protected Resource	Action	Policy Effect	Policy Author	Time State	Logical State	System	Access Subject	Access Resource	Access Action
1	Verify completeness of privacy control permissions		✓	✓		✓		✓				
2	Compare the same user's policy in two similar SNSs (e.g., Facebook vs. Google+) with regards to similar type of protected resources	✓	✓	✓	❋				✓			
3	Compare access subjects authorized by two users for a certain shared protected resource between them	✓			❋	✓						
4	Compare access subjects authorized by a user for a permission between two snapshots of the system (difference due to evolution of user policies)	✓			❋		✓					
5	Compare access subjects authorized for a permission between two snapshots of the system (difference due to evolution of policies)	✓			❋	✓	✓					
6	Compare the protected resources that friends vs. friends-of-friends can see according to a user's policy		✓		❋					✓		

or by observing the system's behavior. Worse is that, since the default settings are not well documented, SNSs can modify them without users noticing it and put them at great risk of privacy violations.

In order to identify the privacy risks users are dealing with in such an above-mentioned ecosystem, we propose an analysis task to formally reason about *completeness* of permissions in an SNS. Our notion of completeness ensures that privacy control policies are applicable to every piece of information related to a user. Therefore, the user can clearly expect to know how her information is protected.

3.3.3.1 Privacy Properties We introduce two properties regarding completeness of privacy control permissions. The target of analysis is the permissions, which can be expressed by partial characterization of policies, as mentioned in Section 3.3.2. We identify three categories of permissions in an SNS:

- *Privacy Setting Permissions*: These are captured by user-configurable privacy settings, which usually have dedicated control elements in the system.
- *Described System Permissions*: These consists of the permissions that are not configurable by users and their corresponding system-defined policies are well documented.
- *Expected Permissions*: These refer to the permissions that a user considers as privacy-sensitive, which may depend on the information model of the SNS and individual expectations.

The privacy setting permissions and described system permissions are disjoint by definition. The expected permissions are subjective in nature. Therefore, depending on the purpose of the analysis, it needs to be determined. Let S, D, and E, represent the classes of privacy setting, described system, and expected permissions in an SNS, respectively. The Venn diagram of Figure 13 is a sample representation of these permission classes. We assume that the expected permissions is likely to cover both privacy setting and described system permissions, i.e., we expect at the least the privacy settings and described system permissions are deemed privacy-sensitive. However, there may exist parts of the expected permissions that are covered by neither privacy settings nor described system permissions.

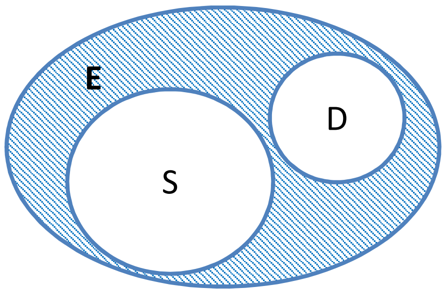


Figure 13: Permission Classes in an SNS: Privacy Settings (S), Described System (D), and Expected (E) Permissions (where E is very likely to cover both S and D)

In other words, there may exist some privacy-sensitive permissions which are not mediated by a transparent policy. The purpose of our analysis task is to reason about existence and nature of such permissions.

We propose two permission completeness properties for an SNS. The following property ensures that users' expected permissions are part of the privacy settings and can be controlled by them.

Definition 3 (Completely Controllable)**.** An SNS with its privacy setting permissions S is *completely controllable* with regards to expected permissions E, if and only if $E \sqsubseteq S$.

Verification of the above property is an analysis task where the subsumption reasoning operation is performed on permissions based on the logical state of *expected* permissions. In practice, with a reasonable assumption about expected permissions, the above notion of completeness cannot be satisfied in SNSs; the users may be overwhelmed by the complexity of the options if they need to control every expected permission. Even if possible, the system provider may choose not to provide such extensive controls due to various design considerations. For example, it might be essential for users to see certain information about their friends in order to keep the friendship relationships valuable in the context of the SNS. We define the following notion of permission completeness as a more practical alternative.

Definition 4 (Completely Known)**.** An SNS with its privacy setting and described system permissions $\langle S, D \rangle$, is *completely known* with regards to expected permissions E, if and only if $E \sqsubseteq (S \text{ or } D)$.

Verification of the above property is an analysis task where two analysis operations are performed. First, the union of the privacy setting and described system permissions are calculated according to policy author analysis factor (Privacy settings are provided by the user and the described system permissions are provided by the system.) Second, based on a logical state analysis factor, the subsumption of expected permissions by the result of the previous step is tested.

3.3.3.2 Case Study: Facebook Privacy Control Permissions In order to demonstrate verification of our permission completeness properties, we analyze the privacy control policies in Facebook. For this purpose, we manually collected the privacy settings and described system permissions from Facebook in December 2012.

Facebook provides a centralized dashboard for controlling privacy settings such as the visibility of the statuses, tags, etc. Moreover, a user can determine the visibility of her profile information such as education, contact info, etc. We collected those settings and formulated their corresponding permissions. In Table 7, we list those settings worded as similarly possible as in the Facebook's privacy settings page, and corresponding permissions. The permission representation is based on partial characterization of policy concept using pResource and pAction properties. For example, in privacy setting permission S2, the first clause specifies the action as insertion. The second complex clause indicates that the resource should be of a reified property class RPannotate, where its object is a Wall, and that wall is owned by me. In other words, the resource is the annotates relationship that annotates a Wall instance belonging to me. Note that the listed privacy settings in Table 7 are limited to the information that can be captured by our proposed ontology in Section 3.1.2. For instance, the setting "*Who can send you Facebook messages?*" is not listed since we did not include messages in our ontology. Our current ontology also does not model third party apps. Moreover, in order to avoid enumerating a tedious list of settings, we provide one privacy setting permission that accounts for all profile attributes (S1). Such a permission specification covers permissions corresponding to settings such as "*Who can look you up using the email address or phone number you provided?*" as well as many other single visibility controls in front of every profile item under a user's *about page*.

Unfortunately, Facebook is not very transparent about its system-defined policies. Some of the system-defined policies are documented in the help pages in the format of questions and answers, which are not easily accessible unless one goes through all such pages. For instance, the following question is provided in the help page for tagging (and not under privacy!): "*When I tag someone in a photo, post or app activity, who can see it?*". The answer provided to this question explains that the audience selected for the post, the user tagged, etc., can see it. Table 8 lists the described system policies that we were able to harvest from Facebook's help pages, and that were related to the information captured in our ontology.

Based on our model of SNS information, our intuition is that a user should be able to control relationships that are about her. In terms of our proposed Facebook ontology, those include properties that directly relate to the user, and the properties that relate to some digital objects owned by the user. We consider these as our expected permissions for a user in an SNS, as shown in Table 9. Note that this is a fairly conservative model of expected permissions in the sense that every piece of information related to a user is considered as potentially privacy-sensitive. However, we see it as a safe and reasonable assumption.

We now present the results of verification of permission completeness properties that were introduced in Section 3.3.3.1 with regards to the permissions presented in Tables 7, 8, and 9. According to the listed permissions, it is not very surprising that Facebook does not satisfy the *completely controllable* property (Definition 3). However, Facebook does not satisfy the *completely known* (Definition 4) property either. It means that there exist expected permissions that are neither in the privacy settings nor in the described system permissions. In order to identify such missing permissions, we formulated several permission classes (subclasses of expected permissions) M_i, and tested them for subsumption satisfiability according to $\mathsf{M}_i \sqsubseteq \mathsf{E} - (\mathsf{S} \text{ or } \mathsf{D})$. M_i is an expected permission that is neither in S nor D if the subsumption relationship is satisfied. We report in Table 10 some permissions that are missing in current Facebook privacy control policies according to our analysis. Note that the table lists only a sample of permissions with selection as their action; many other missing permissions can be formulated for insertion and deletion actions.

Table 7: Privacy Setting Permissions in Facebook

	Privacy Setting/Corresponding Permission
S1	Profile attributes privacy (for each item) pAction value pa_select and pResource some (rdpSbj value me)
S2	Who can post on your timeline? pAction value pa_insert and pResource some (RPannotates and (ropObj some (Wall and (inverse ropObj some (RPowns and (ropSbj value me))))))
S3	Who can see what others post on your timeline? pAction value pa_select and pResource some (RPannotates and (ropObj some (Wall and (inverse ropObj some (RPowns and (ropSbj value me))))) and (ropSbj some (WallPost and (inverse ropObj some (RPowns and (ropSbj value me))))))
S4	Who can see what you post? (per item) pAction value pa_select and pResource some (RPhasContent and (rdpSbj some (inverse ropObj some (RPowns and (ropSbj value me)))))
S5	Review tags friends add to your own posts pAction value pa_insert and pResource some (RPannotates and (ropObj some (inverse ropObj some (RPowns and (ropSbj value me)))) and (ropSbj some PhotoUserTag))
S6	Visibility of photos (managed per album) pAction value pa_select and pResource some (RPhasContent and (rdpSbj some (inverse ropObj some (RPowns and (ropSbj value me))))) or (RPowns and (ropSbj value me) and (ropObj some Photo))

Table 8: Described System Permissions in Facebook

	Described System Policy/Corresponding Permission
D1	Who can see tags I make? pAction value pa_select and pResource some (RPannotatesWith and (ropSbj some (UserTag and (inverse ropObj some (RPowns and (ropSbj value me))))))
D2	Who can see that I'm tagged in a post? pAction value pa_select and pResource some (RPannotatesWith and (ropObj value me))
D3	Who can see a tag that someone added to your post? pAction value pa_select and pResource some (RPannotates and (ropObj some (UserTaggable and (inverse ropObj some (RPowns and (ropSbj value me))))))

Table 9: Expected Permissions for Facebook

	Expected Privacy Setting/Corresponding Permission
E1	Control whatever relates to you pAction some Action and pResource some (ReifiedProperty and ((rdpSbj value me) or (ropObj value me) or (ropSbj valueuj me)))
E2	Control whatever relates to something belonging to you pAction some Action and pResource some (ReifiedProperty and ((rdpSbj some (inverse ropObj some (RPowns and (ropSbj value me)))) or (ropObj some (inverse ropObj some (RPowns and (ropSbj value me)))) or (ropSbj some (inverse ropObj some (RPowns and (ropSbj value me))))))

Table 10: Sample Missing Permissions in Facebook

	Missing Policy/Corresponding Permission
M1	Who can see that you have tagged someone? pAction value pa_select and pResource some (RPowns and (ropObj some PhotoUserTag) and (ropSbj value me))
M2	Who can see that you have liked something? pAction value pa_select and pResource some (RPlikes and (ropSbj value me))
M3	Who can see your comment on someone else's post? pAction value pa_select and pResource some (RPannotates and (ropSbj some (Comment and (inverse ropObj some (RPowns and (ropSbj value me))))) and (ropObj some (inverse ropObj some (RPowns and (ropSbj some Others))))
M4	Who can see if you are friend with someone? pAction value pa_select and pResource some (RPisFriendOf and (ropSbj value me))

4.0 A FRAMEWORK FOR STRUCTURE PRESERVING ANONYMIZATION OF SOCIAL NETWORKS

In this chapter, we present our framework to apply structure-preserving heuristics to existing edge-perturbation anonymization schemes for social networks. We empirically show that our proposed enhancement methods can preserve structural properties of social networks significantly better compared to what original edge perturbation algorithms offer. For this purpose, we present results of experiments on three different enhanced anonymization algorithms and multiple datasets, analyzing social network analysis measures such as betweenness centrality, clustering coefficient, etc.

The rest of the chapter is organized as follows. In Section 4.1, we provide an abstract representation of edge-perturbing anonymization algorithms. In Section 4.2, we define the notions of structural roles in social networks and role dissimilarity, and present an algorithm to calculate them for undirected networks. We propose our approaches to preserve structural properties by using concepts of roles and edge betweenness in Sections 4.3 and 4.4, respectively. We evaluate the proposed approaches using multiple datasets and various evaluation metrics in Section 4.5. In Section 4.6, we discuss privacy implications of our approach and a practical way to select anonymization enhancement parameters.

4.1 EDGE-PERTURBING ANONYMIZATION

An undirected social network is defined as a graph, $G\langle V, E\rangle$, where the set of vertices V represents the agents in the network, and the set of undirected edges $E \subseteq \{(u, v)|u, v \in V\}$ represents the relationships between agents in V. Edge-perturbing anonymization techniques modify edges of a network to satisfy a certain *anonymization criteria*. These techniques typically follow a greedy iterative approach, which can be abstractly expressed as in Algorithm 1. In each iteration, Algorithm 1 selects an edge to be added/removed using a heuristic which

Algorithm 1 Iterative Edge Perturbation

```
Require: $G\langle V, E\rangle$
Ensure: Anonymized version of $G\langle V, E\rangle$
 1: repeat
 2:   if an edge should be added then
 3:     Choose non-existent edge $(u, v)$ to be added
 4:     $E \leftarrow E \cup \{(u, v)\}$
 5:   if an edge should be removed then
 6:     Choose existing edge $(u, v)$ to be removed
 7:     $E \leftarrow E \backslash \{(u, v)\}$
 8:   if anonymization criteria is not achievable then
 9:     return null
10: until anonymization criteria is achieved
11: return $G\langle V, E\rangle$
```

depends on the specific technique. The iterations continue until the graph is considered anonymized according to the anonymization criteria. Different anonymization techniques have different anonymization criteria. In the *random perturbation* technique [32], the goal is to simply remove m edges randomly and then add m random edges. In the *k-anonymity-based* approaches (e.g., [53, 83, 76, 33]), the goal is, for instance, to achieve a graph with k-anonymous vertex degrees (e.g., `Supergraph` [53]). The algorithm aborts if the anonymization criteria cannot be achieved, which is also dependent on the actual technique.

The `Greedy-Swap` algorithm proposed in [53] includes an optimization phase to select a group of edge changes in the graph in each iteration, which results in a slightly different algorithm scheme (see Algorithm 2). The algorithm first creates an anonymized random graph based on a k-anonymous degree sequence of the original graph. In each iteration, every pair of edges in a subset of existing edges is examined for a *swap*. In a *swap* operation, a pair of edges are replaced with another pair using the same end nodes. Two swap options

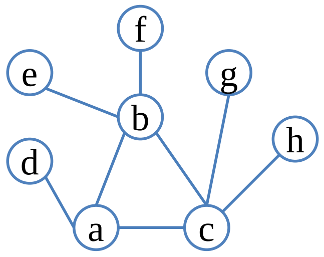
Algorithm 2 `Greedy-Swap`

```
Require: $G\langle V, E\rangle$
Ensure: Anonymized version of $G\langle V, E\rangle$
1: Create anonymized random social network $G'\langle V, E'\rangle$, where $|E'| = |E|$
2: repeat
3:   Select $\log(|E'|)$ of edges $E'$ randomly
4:   for all pairs of selected edges $(u, v)$ and $(u', v')$ do
5:     Calculate the gain value considering swapping of the pair either with $(u, u')$ and $(v, v')$, or $(u, v')$ and $(u', v)$
6:   Perform the swap with maximum gain (if any)
7: until no edge swap is performed
8: return $G'\langle V, E'\rangle$
```

are considered for a pair of edges $\{(u, v), (u', v')\}$: either $\{(u, u'), (v, v')\}$, or $\{(u, v'), (u', v)\}$. Such swaps do not change vertex degrees, thus, ensuring the already-established degree k-anonymity. A *gain value* is calculated for each swap option, and the swap with maximum (positive) gain is selected. In [53], the authors calculate the *gain value* as the increment of edge overlap (intersection) between the interim and the original graph. Performing the swap with maximum gain at each iteration would greedily make the anonymized graph more structurally similar to the original one.

4.2 STRUCTURAL ROLES

There are three major approaches to classify agents in a network based on relationships among them into their social positions: *structural*, *automorphic*, and *regular equivalence* [78, 45]. Two agents are *structurally equivalent* if they have identical ties with identical other agents. In *automorphic equivalence*, agents in the same position must have identical ties with different sets of agents that play the same role in relation to that position. Finally, two agents are *regularly equivalent* if they have same kind of relationships with agents that



structural:

{{a},{b},{c},{d},{e, f},{g,h}}

automorphic:

{{a},{b,c},{d},{e,f,g,h}}

regular:

{{a,b,c},{d,e,f,g,h}}

Figure 14: Sample Equivalency Classes for a Network

are also regularly equivalent. Figure 14 shows a small example of equivalency classes based on each of these concepts. We choose regular equivalence among these as it captures the concept of structural roles very well, and is the least restrictive among the three concepts.

In this section, we formally define the notion of roles in the context of undirected social networks, adopting some definitions from [50]. We also define the extent of regular equivalence between roles and introduce an algorithm for identifying roles and calculating such a measure.

4.2.1 Roles based on Regular Equivalence

Definition 5 (Role Assignment)**.** A role assignment for network $G\langle V, E\rangle$ is a function $\Phi : V \to R$, defined for every member of V, where R is a set of roles.

A role assignment partitions agents into equivalency classes. Two agents are considered *equivalent* ($\equiv_\Phi$) if they are assigned the same role: $\forall u, v \in V; u \equiv_\Phi v \Leftrightarrow \Phi(u) = \Phi(v)$. In other words, a role assignment is a projection of an equivalence relation. Of our particular interest is the *regular equivalence*. The following definition captures the relationships between agents.

Definition 6 (Neighbor Role Set)**.** $\Gamma_\Phi : V \to 2^R$ is a function that maps an agent in network $G\langle V, E\rangle$ to the roles of its neighbors according to role assignment Φ, i.e., $\Gamma_\Phi(u) =$

$\{\Phi(v)|(u,v) \in E\}$.

We recall that regularly equivalent agents must have the same kind of relationships with other regularly equivalent agents. Therefore, we define a role assignment that projects a regular equivalence relation as follows.

Definition 7 (Regular Equivalence Role Assignment)**.** A role assignment $\Phi : V \rightarrow R$ projects a regular equivalence for agents in network $G\langle V, E\rangle$ iff

$$\forall u, v \in V, \Phi(u) = \Phi(v) \Rightarrow \Gamma_\Phi(u) = \Gamma_\Phi(v).$$

Two agents are regularly equivalent if and only if they have neighbors with the same roles. We refer to this as *RE-role assignment*.

4.2.2 Extent of Role Equivalence

In the case that two agents are not regularly equivalent, it is sometimes desirable to know to what extent they are playing similar roles. That is particularly useful in our enhancement approach, which relies highly on the existence of large classes of equivalent agents. As per our experiments, algorithms for computing regular equivalence usually result in low-populated equivalency classes. Therefore, as we discuss in later sections, agents playing similar roles can be considered as alternatives to the non-existing regularly equivalent agents. We abstractly define a dissimilarity measure for roles as follows.

Definition 8 (Regular Equivalence Role Dissimilarity)**.** $\Delta_\Phi : V \times V \rightarrow [0, 1]$ is a role dissimilarity function for agents of network $G\langle V, E\rangle$ corresponding to role assignment Φ, where, at the two extremes, $\Delta_\Phi(u, v) = 0$ implies agents u and v have the same role ($\Phi(u) = \Phi(v)$), and $\Delta_\Phi(u, v) = 1$ implies agents u and v have completely dissimilar roles.

The role dissimilarity measure is usually dependent on the regular equivalence computation scheme in use. We provide the details in Section 4.2.3. Subsequently, we are interested in a dissimilarity measure between two sets of roles, given the dissimilarity measure for individual pairs of roles. The rationale behind computing such a dissimilarity measure is to see how similar a modified neighbor role set of an agent is to its original one if some of its ties are changed. We define this measure as follows.

Definition 9 (Regular Equivalence Role Set Dissimilarity)**.** Let $S, S' \subseteq R$ be two subsets of roles. The regular equivalence dissimilarity between S and S', written as $\Lambda(S, S')$, is equal to:

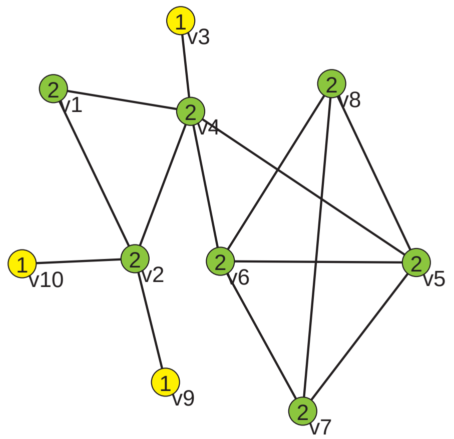
$$\frac{\frac{\sum_{x \in S} \sqrt[|S'|]{\prod_{y \in S'} \Delta(x,y)}}{|S|} + \frac{\sum_{y \in S'} \sqrt[|S|]{\prod_{x \in S} \Delta(x,y)}}{|S'|}}{2}$$

The above formula essentially calculates the (asymmetric) dissimilarities of S to S', and S' to S, and then takes the average to compute an overall (symmetric) dissimilarity between S and S'. The dissimilarity of S to S' (the first expression in the numerator) is calculated as follows. For every role x in S, the product of its dissimilarities with all roles in S' is calculated, and its $|S'|^{\text{th}}$ root is taken. This gives us an overall dissimilarity value between x and roles in S'. If one of the roles in S' is the same as x, the result would be zero; otherwise, the dissimilarity values for each will be effective in the result. The average of all such dissimilarities for all the roles in S is considered as the dissimilarity of S to S'. The dissimilarity of S' to S is calculated in a similar fashion.

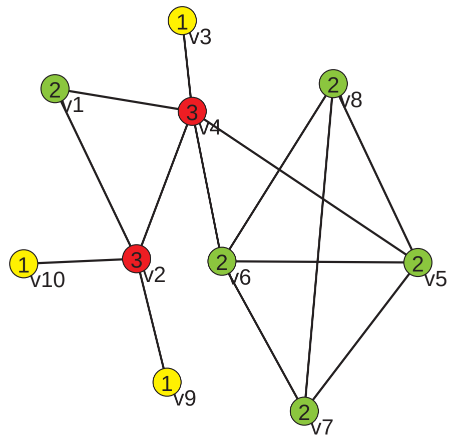
4.2.3 An Algorithm to Identify Roles

REGE is a simple algorithm to partition agents based on regular equivalence. However, as pointed out by Borgatti and Everett [9], there are some inconsistencies with the algorithm in recognizing regular equivalence partitions. Also, there are issues related to the similarity measure it generates such as being affected by degree of nodes (which theoretically should not occur because of the nature of regular equivalence). CATREGE [9] is an alternative solution for finding regular equivalence in categorical network data, i.e., networks with different types of edges. It also works for non-categorical data that is the concern of our work. Moreover, the similarity measure computed by CATREGE avoids the above-mentioned issues.

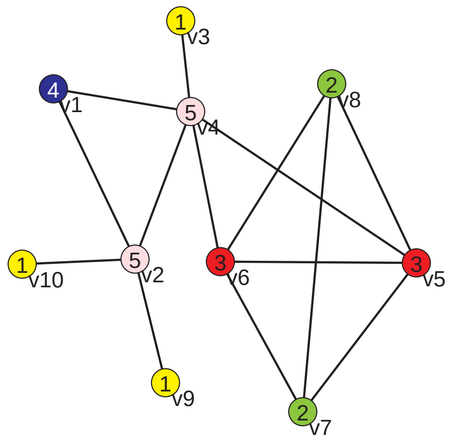
Since we deal with non-categorical (single-type edge), undirected social networks employing CATREGE results into an uninteresting regular equivalence: all agents will be classified in a same equivalency class. We modify the CATREGE algorithm to tackle this issue, as shown in Algorithm 3. We initialize our algorithm with two partitions of equivalent agents: agents with the minimum degree (disregarding isolates) form one partition, and the rest of the agents form the other partition. In each iteration, the algorithm checks if pairs of



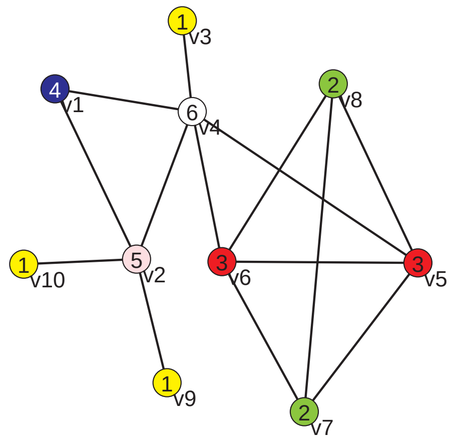
(a) Initial State



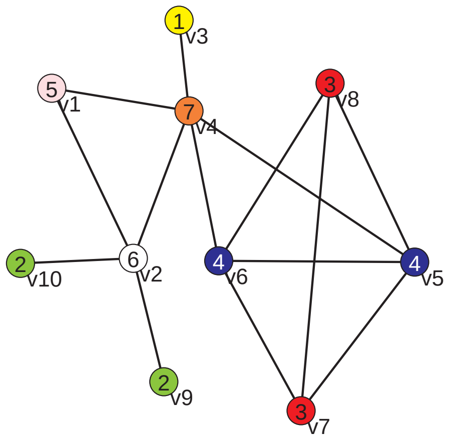
(b) Iteration 1



(c) Iteration 2



(d) Iteration 3



(e) Iteration 4

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
v_1	5	1	0	1	2	2	2	2	0	0
v_2	1	5	0	3	1	1	1	1	0	0
v_3	0	0	5	0	0	0	0	0	4	4
v_4	1	3	0	5	1	1	1	1	0	0
v_5	2	1	0	1	5	5	2	2	0	0
v_6	2	1	0	1	5	5	2	2	0	0
v_7	2	1	0	1	2	2	5	5	0	0
v_8	2	1	0	1	2	2	5	5	0	0
v_9	0	0	4	0	0	0	0	0	5	5
v_{10}	0	0	4	0	0	0	0	0	5	5

(f) (Non-Normalized) Similarity

Partition Color Codes: 1=Yellow, 2=Green, 3=Red, 4=Purple, 5=Pink, 6=White, 7=Orange

Figure 15: Sample Execution of Algorithm 3

Algorithm 3 Calculate Agents' Role Dissimilarities

```
1:  $p_1 \leftarrow \{u \in V | degree(v) = Min(degree(v_i \in V))\}$
2:  $p_2 \leftarrow V \setminus p_1$
3:  $P \leftarrow \{p_1, p_2\}$
4:  Initialize $\Phi$ according to partition set $P$:
    $\forall u \in V[u \in p_i \rightarrow \Phi(u) = r_i]$
5:  $i \leftarrow 1$
6:  repeat
7:    for each partition $p \in P$ do
8:      Split $p$ into independent partitions $\{p_i\}$ such that $\forall u, v \in p\ [u, v \in p_i \leftrightarrow \Gamma_\Phi(u) = \Gamma_\Phi(v)]$
9:      if $\Gamma_\Phi(u) \neq \Gamma_\Phi(v)$ then
10:       $Similarity(u, v) \leftarrow i$
11:     Substitute $p$ in $P$ with partitions $\{p_i\}$
12:   Update $\Phi$ according to partition set $P$
13:   $i \leftarrow i + 1$
14: until no changes in partitions set $P$
15: for every pair of agents $u$ and $v$ do
16:   $\Delta_\Phi(u, v) \leftarrow (i - Similarity(u, v))/i$
```

nodes that were equivalent in the previous iteration are connected to other agents that were equivalent themselves. If not, they are marked as non-equivalent. The procedure is repeated until there is no change in the equivalencies compared to the previous iteration. The extent of regular equivalence between two agents can be obtained by counting the number of iterations it takes them to split into different partitions. The algorithm obtains a normalized dissimilarity by subtracting this value from and dividing it by the total number of iterations. A naive implementation of Algorithm 3 has time complexity $O(dn^3)$, where n and d are node count and maximum node degree of the input network. However, in practice, the algorithm converges much sooner than the worst case complexity indicates.

Algorithm 3 is different from the original CATREGE in two aspects. First, it does not deal with multiplex matrix required for categorical data. Second, it begins with a specific initial partitioning, as opposed to all agents being in the same partition in the original CATREGE. Our initial partitioning essentially indicates that peripheral agents in a network are more regularly equivalent to each other, and less so with the other agents that fall inside the network.

Figure 15 illustrates the execution of Algorithm 3 on a small network. In each iteration, nodes within a same partition are marked with a same color (number). Note that partition colors (numbers) only indicate equivalent agents in one iteration and do not carry any other semantics. In the initial state (Figure 15a), vertices v_3, v_9, and v_{10} are colored yellow, and all the others are colored green. Figure 15b illustrates the resultant partitions after the first iteration. Since in the previous step, the yellow vertices were all connected to the green vertices, they will not separate in this iteration. However, the previously green vertices are divided into two partitions: the ones that were only connected to greens, and the ones that were connected to both yellows and greens. If we continue the procedure, the final result is obtained after iteration 4 (Figure 15e); further iterations will not change the partitions. Figure 15f shows the (non-normalized) extent of regular equivalence between agents. For instance, the similarity value between v_1 and v_2 is 1, because they were separated after the first iteration. Analogously, the similarity value between v_5 and v_7 is 2, because they were separated after the second iteration. If two vertices eventually remain equivalent, their similarity value will be the maximum number of steps (e.g., 5 for v_5 and v_6). These values are converted to dissimilarity measure in the last loop of Algorithm 3.

4.3 PRESERVING STRUCTURAL PROPERTIES USING ROLES

Our intuition is that preserving the role structure of a social network in the anonymization process would preserve to some extent the network structural properties such as centralities. Therefore, the anonymized network will be more suitable to be used for typical network analysis. To this end, we need to ensure that an RE-role assignment in the original network is applicable to its edge-perturbed version as well. In this section, we present our proposed approach that preserves role structure, and apply it on the edge perturbation algorithms presented in Section 4.1.

Preserving an RE-role assignment while perturbing a social network is not straightforward. Because modifications to the edge structure of a network during perturbation and changing neighborhoods of agents can easily invalidate an RE-role assignment for the edge-perturbed version of a network. The following theorem captures a sufficient condition to ensure that.

Theorem 3. *Let $G'\langle V, E'\rangle$ be an edge-perturbed version of network $G\langle V, E\rangle$. An RE-role assignment Φ for G is also an RE-role assignment for G' if*

$$\forall u \in V \ [\Gamma_{\Phi}^{G'}(u) = \Gamma_{\Phi}^{G}(u)]$$

Proof. The proof follows from the above condition and Definition 7. Given Φ is an RE-role assignment for G, for a pair of agents u and v where $\Phi(u) = \Phi(v)$, by Definition 7, we have $\Gamma_{\Phi}^{G}(u) = \Gamma_{\Phi}^{G}(v)$. Now, if the condition in the theorem is true, we also have $\Gamma_{\Phi}^{G'}(u) = \Gamma_{\Phi}^{G}(u)$ and $\Gamma_{\Phi}^{G'}(v) = \Gamma_{\Phi}^{G}(v)$. Based on these three equalities, we have $\Phi(u) = \Phi(v) \Rightarrow \Gamma_{\Phi}^{G'}(u) = \Gamma_{\Phi}^{G'}(v)$, which is a necessary and sufficient condition for Φ to be an RE-role assignment for G', according to Definition 7. $\square$

Theorem 3 states that keeping the neighbor role sets of agents in a network intact in the anonymization process will preserve an RE-role assignment. As an edge perturbation algorithm involves a series of edge additions/removals, the above condition can be further elaborated with regards to the set of added or removed edges as in the following theorem.

Theorem 4. *Let $G'\langle V, E'\rangle$ be an edge-perturbed version of network $G\langle V, E\rangle$. An RE-role assignment Φ for G is also an RE-role assignment for G' if the following conditions are met*

$$\forall (u,v) \in E_i \ \exists (u,v') \in E \ [\Phi(v) = \Phi(v')] \tag{4.1}$$

$$\forall (u,v) \in E_d \ \exists (u,v') \in E' \ [\Phi(v) = \Phi(v')] \tag{4.2}$$

where sets $E_i = E' \backslash E$ and $E_d = E \backslash E'$ represent added and removed edges, respectively.

Proof. Since the same role assignment Φ is considered for both G and G', any difference between $\Gamma_\Phi^G(u)$ and $\Gamma_\Phi^{G'}(u)$, for any agent u, can only be the result of either addition or removal of an edge adjacent to u. For an added edge (u, v), by condition (4.1), we have $\exists(u,v') \in E[\Phi(v) = \Phi(v')]$ and therefore $\Phi(v) = \Phi(v') \in \Gamma_\Phi^G(u)$, i.e., an added edge would not affect the neighbor role set of an agent. For a removed edge $\langle u, v\rangle$, by condition (4.2), we have $\exists(u,v') \in E'[\Phi(v) = \Phi(v')]$ and therefore, $\Phi(v) = \Phi(v') \in \Gamma_\Phi^{G'}(u)$, i.e., a removed edge would not affect the neighbor role set of an agent. These suggest

$$\forall u \in V \ [\Gamma_\Phi^{G'}(u) = \Gamma_\Phi^G(u)]$$

which is a sufficient condition for Φ to be an RE-role assignment for G', according to Theorem 3. □

4.3.1 Role-Enhanced Iterative Edge Perturbation

Based on Theorem 4, we extend and enhance the iterative edge perturbation techniques represented by Algorithm 1 as follows. After selecting an edge for addition, it is added only if it conforms to condition (4.1). For this purpose, line 4 of the algorithm should be replaced with the following.

```
**if** $\exists(u,v'),(u',v) \in E \ [\Phi(v) = \Phi(v') \wedge \Phi(u) = \Phi(u')]$ **then**
    $E \leftarrow E \cup \{(u,v)\}$
```

This checks if there exists vertex v' in u's neighborhood with the same role as v's, and if there exists vertex u' in v's neighborhood with the same role as u's. If either of the checks fails the edge addition is discarded. Analogously, an edge removal should be allowed only if it conforms to condition (4.2). As per Theorem 4, such a modified version of Algorithm 1 will preserve an RE-role assignment for the graph in each iteration. Therefore, an RE-role assignment for the original social network graph will be valid for its final edge-perturbed version.

Although theoretically sound, the above-mentioned strategy may not perform well in practice. Based on our experiments on network datasets, algorithms such as the one presented in Section 4.2.3 identify very small number of agents with the same role. Therefore, when adding/removing an edge, e.g., (u, v), the chance of finding an agent with the same role as v's in u's neighborhood is very low, and vice versa. The above-mentioned strategy is hard to be applied in such a situation as it would reject changes to the network, because of low population of equivalent agents in every class. In order to overcome this limitation, we use a relaxed version of the conditions in Theorem 4. Instead of an exact role match as proposed in the conditions, we propose a partial match by using a threshold on RE-role dissimilarity between agents. Algorithm 4 provides the pseudocode for the enhanced version of the iterative edge perturbation approach. As the input arguments, it requires role dissimilarity values for agents (Δ_Φ) and threshold $\delta \in [0, 1]$ which indicates the extent of non-perfect role matching to be allowed. The time complexity of Algorithm 4 is clearly dependent on the actual iterative edge perturbation method. However, it will be bounded by $O(n^2)$ since in the worst case all the candidate edges for addition would be tested.

The following example demonstrates how maintaining the neighbor role set of an agent can help in preserving structural properties of a network.

Example 5. In Figure 15e, assume we need to remove one of the adjacent edges to v_4. Based on our proposed scheme, an agent should exist in v_4's neighborhood that has low role dissimilarity with the agent that is removed from that neighborhood, and vice versa. Note that we can instead consider high similarity values in Figure 15f. The following list shows the most similar agent in v_4's neighborhood after removing each candidate from it.

Algorithm 4 Role-Enhanced Iterative Edge Perturbation

```
Require: $G\langle V, E\rangle$, $\Delta_\Phi$, and $\delta$
Ensure: Anonymized version of $G\langle V, E\rangle$
 1: repeat
 2:   if an edge should be added then
 3:     Choose non-existent edge $(u, v)$ to be added
 4:     if $\exists (u, v'), (u', v) \in E\ [\Delta_\Phi(v, v') < \delta \wedge \Delta_\Phi(u, u') < \delta]$ then
 5:       $E \leftarrow E \cup \{(u, v)\}$
 6:   if an edge should be removed then
 7:     Choose existing edge $(u, v)$ to be removed
 8:     $E' \leftarrow E \setminus \{(u, v)\}$
 9:     if $\exists (u, v'), (u', v) \in E'\ [\Delta_\Phi(v, v') < \delta \wedge \Delta_\Phi(u, u') < \delta]$ then
10:       $E \leftarrow E'$
11:   if anonymization criteria is not achievable then
12:     return null
13: until anonymization criteria is achieved
14: return $G\langle V, E\rangle$
```

- v_1: v_5/v_6 with similarity value 2.
- v_2: $v_1/v_5/v_6$ with similarity value 1.
- v_3: No similar agent exists.
- v_5: v_6 with similarity value 5.
- v_6: v_5 with similarity value 5.

The following list shows the most similar agent in each of the above candidate's neighborhood that may replace v_4's role.

- v_1: v_2 with similarity value 3.
- v_2: v_1 with similarity value 1.
- v_3: No other neighbor agent exists.
- v_5: $v_6/v_7/v_8$ with similarity value 1.
- v_6: $v_5/v_7/v_8$ with similarity value 1.

As suggested by the above similarities, edge (v_1, v_4) seems to be the best option to remove. Because both v_1 and v_4 have an agent in their neighborhood with moderate similarity to the other. To ensure this is indeed the best choice, we calculate two network measures, i.e, mean betweenness and closeness centralities, for the result of each case. The mean betweenness (closeness) centrality for the original network is 5.1 (0.487), and is as follows after removing each of the candidates:

- v_1 : 5.7 (0.458)
- v_2 : 6.9 (0.411)
- v_3 : 3.9 (0.318)
- v_5 : 5.7 (0.456)
- v_6 : 5.7 (0.456)

The above centrality values confirms our choice of removing (v_1, v_4), since the corresponding resulting network has closer centrality values to the original network compared to the other choices.

4.3.2 Role-Enhanced `Greedy-Swap`

As mentioned in Section 4.1, the `Greedy-Swap` algorithm follows a different overall procedure than most of the other perturbation approaches. Hence, we need a different approach to enhance it for preserving role structure. We do so by proposing a new *gain function* in Algorithm 2. Recall that the `Greedy-Swap` technique [53] starts with an anonymized version of the network but with randomized edges, and performs edge swaps to make the anonymized network as similar as possible to the original graph. To this end, the authors define the gain measure as the increase in the edge overlap between the original and the anonymized networks. We propose to substitute the *gain function* in Algorithm 2 with a *role similarity gain* measure which is calculated based on regular equivalence role structure. The role similarity gain function measures how much each of the involved vertices in an edge swap gets closer (more similar) to its corresponding original state in terms of the role structure. Recall from Theorem 3 that the neighbor role set of an agent acts as an important factor in preserving its role. Hence, we consider it as the main clue for calculating such a role similarity gain.

Let u be an agent involved in an edge swap, and $\Gamma_\Phi^G(u)$ be its neighbor role set in the original network. Also, in the i^{th} iteration of Algorithm 2, let $\Gamma_\Phi^{G_i}(u)$ be its neighbor role set in the interim network, and $\Gamma_\Phi^{G_{i+1}}(u)$ be its neighbor role set in the next state of the interim network if the swap is performed. The objective of optimization based on role similarity gain is to obtain better similarity between $\Gamma_\Phi^{G_{i+1}}(u)$ and $\Gamma_\Phi^G(u)$ compared to $\Gamma_\Phi^{G_i}(u)$ and $\Gamma_\Phi^G(u)$. In other words, using dissimilarity measures defined in Section 4.2.2, we need to have

$$\Lambda(\Gamma_\Phi^G(u), \Gamma_\Phi^{G_i}(u)) \geq \Lambda(\Gamma_\Phi^G(u), \Gamma_\Phi^{G_{i+1}}(u))$$

Hence, the role similarity gain can be measured by the *decrease in dissimilarity* between the neighbor role sets in the original and interim networks. The bigger the gap between the two sides of the above inequality, the larger the role similarity gain will be. As there are four vertices involved in a swap of a pair of edges (u, v) and (u', v'), we calculate the total role similarity gain of a swap as follows.

$$\frac{\sum_{x \in \{u,v,u',v'\}} [\Lambda(\Gamma_\Phi^G(x), \Gamma_\Phi^{G_i}(x)) - \Lambda(\Gamma_\Phi^G(x), \Gamma_\Phi^{G_{i+1}}(x))]}{4}$$

The time complexity is $O(\log^2 n)$ for Algorithm 2, and $O(d^2)$ for the above gain function, where n and d are node count and maximum degree of the input network, respectively. Therefore, the time complexity for role-enhanced `Greedy-Swap` is $O(d^2 \log^2 n)$.

4.4 PRESERVING STRUCTURAL PROPERTIES BASED ON EDGE BETWEENNESS

Computing shortest paths between pairs of nodes in a network is an underlying factor for social network analysis measures, ranging from simple graph-level measures such as characteristic path length (average path length between node pairs) and diameter (maximum shortest path in the network) to node level centrality measures such as betweenness (proportion of shortest paths that pass through a node) and closeness (average distance of a node to all the other nodes). In this section, we propose an algorithm to maintain structural properties in a perturbed network by limiting the amount of changes to the shortest paths in the network.

We leverage the notion of *edge betweenness* to control the shortest paths, which was introduced in the Newman-Girvan community detection algorithm [64]. Edge betweenness is defined for an edge as the number of shortest paths between any pair of nodes that pass through that edge. If there are more than one shortest paths for a pair of nodes, they are counted proportionally so that they sum up to unity. Intuitively, based on the definition, if an edge has a low edge betweenness centrality removing/adding it from/to the network will have less effect on the shortest paths compared to removing/adding an edge with higher betweenness. A lesser number of shortest path changes in the network due to such edge addition/removal would help to have less change in social network analysis measures such as closeness. However, note that although this strategy limits the number of shortest path changes, it cannot control the amount of change in the shortest paths. This observation is central to our proposed approach to perturb a network with limited changes to shortest paths.

Algorithm 5 provides the pseudocode for the enhanced version of the iterative edge

Algorithm 5 Edge-Betweenness-Enhanced Iterative Edge Perturbation

```
Require: $G\langle V, E\rangle$ and $\beta$
Ensure: Anonymized version of $G\langle V, E\rangle$
 1: repeat
 2:   if an edge should be added then
 3:     Choose non-existent edge $(u, v)$ to be added
 4:     $E' \leftarrow E \cup \{(u, v)\}$
 5:     if Normalized-Edge-Betw$_{\langle V,E'\rangle}((u, v)) < \beta$ then
 6:       $E \leftarrow E'$
 7:   if an edge should be removed then
 8:     Choose existing edge $(u, v)$ to be removed
 9:     if Normalized-Edge-Betw$_{\langle V,E\rangle}((u, v)) < \beta$ then
10:       $E \leftarrow E \setminus \{(u, v)\}$
11:   if anonymization criteria is not achievable then
12:     return null
13: until anonymization criteria is achieved
```

perturbation anonymization using edge-betweenness. Here, function *Normalized-Edge-Betw* calculates the betweenness of an edge and normalizes it based on the maximum edge betweenness value of the edges in the graph. $\beta \in [0, 1]$ is an input argument that limits the potential set of edges to be added/removed, based on their normalized edge betweenness centrality value. Edge-betweenness calculation has time complexity $O(ne)$, where n and e are node and edge count, respectively. Therefore, a naive implementation of the algorithm will have time complexity $O(n^3e)$.

4.5 EXPERIMENTS

4.5.1 Experimental Setup

In order to evaluate the performance of the proposed enhanced approaches in terms of preserving data utility, we set up an extensive experimental study. In this section, we describe our choices for evaluation measures, datasets, and implementing anonymization algorithms.

4.5.1.1 Evaluation Measures Since we do not focus on anonymization for a specific application purpose it is hard to identify exact representative measures of data utility. In the area of k-anonymization for tabular data [16], the basic two actions to perform on a dataset are generalization and suppression. In generalization, attribute values of tuples are replaced by more general values. In suppression, tuples may be removed from the dataset (when generalization is not sufficient). The proposed algorithms for tabular data aim to minimize such generalization and suppression. This can generally be computed based on a formula of information loss for each tuple. However, it is much more challenging to consider such a measure for social networks [32, 82, 79]. Perturbation of social networks, even when limited to modifying edges, can affect many node and network properties. In other words, the effect of changes in the graph is not local (as it was for tuples in the case of tabular data). For this reason, instead of considering a single measure, we study the preservation of a set of social network analysis measures. We have selected two widely used measures at network level [78, 63], namely, average path length and clustering coefficient. Also, since studying node centrality in a network is of prime importance, we have considered a number of node level centrality measures based on previous related studies on the quality of social network data [8, 26]. We describe the set of measures in the following list:

- *Average Path Length* is the average distance of all the node pairs in the network.
- *Clustering Coefficient* measures the tendency of nodes in a network to cluster together, by counting the ratio of closed triplets to connected triplets in the network.
- *Betweenness Centrality* is the number of times a node falls on the shortest paths between node pairs in the network. We consider the average of this value for all the nodes in the network.
- *Closeness Centrality* is the average distance of a node to all the nodes in the network. We consider the average of this value for all nodes in the network.
- *Centrality Rankings:* We note that it is important in social network analysis to identify the most central nodes in a network. Therefore, in addition to the mean centrality values we investigate the accuracy of preserving individual nodes' centrality rankings. For this purpose, we consider the function of an anonymization algorithm as a classifier that should classify the top n central nodes in the original network as the most central nodes. We evaluate the performance of such a classifier by measuring the area under the ROC curve (AUC). The accuracy measures *Top3AUC* and *Top10pAUC*, respectively, indicate such performance metric for classifying the top three and the top decile of central nodes in the original network as the most central in the anonymized network. The accuracy metrics are considered separately for betweenness and closeness centrality.
- *Centrality Correlation* (R^2) measures the square of Pearson correlation between centralities of nodes in the original and perturbed networks, as an alternative accuracy metric.

The effectiveness of the proposed approaches is evaluated as follows. We calculate the above-mentioned measures on the outputs of both original anonymization algorithms and their enhanced versions, and compare them with the corresponding measurements on the original network. An anonymized output that provides closer measurements to those of the original network is intuitively more useful for analysis.

4.5.1.2 Datasets Since the network measures are dependent on the network structure, it is common in the study of social networks to experiment on several networks [51, 44, 8]. In our study, we experiment on both real-world and synthetic datasets. We have selected two published real-world datasets in the literature, and have generated three synthetic social networks based on different topological models [63]. The social network datasets in our experiments are as follows.

- `PolBooks`: A network of books about US politics sold by Amazon.com around the 2004 presidential election (compiled by V. Krebs, *www.orgnet.com*). Edges between books represent their frequent purchase by the same buyers.
- `Jazz`: A collaboration network of jazz musicians [27], where nodes represent bands and edges indicate that the corresponding bands share a common musician.
- `ER`: A synthetic random network based on Erdos-Renyi model.
- `BA`: A synthetic scale-free network based on Barabasi's model.
- `SW`: A synthetic small-world network.

Table 11 lists some of the structural properties of our datasets. Since the original anonymization algorithms that we are enhancing are not fast algorithms we choose small size datasets in our experiments. Note that the sizes of datasets are not an influencing factor on how our approach is successful in achieving its goal to preserve data utility.

4.5.1.3 Implemented Algorithms We implemented the original, role-enhanced, and edge-betweenness-enhanced versions of random perturbation [32], `Supergraph` [53], and `Greedy-Swap` [53]. The first two algorithms are variations of the iterative edge perturbation approach, as described in Section 4.1. The random perturbation algorithm first removes m edges from a network and then adds m other edges at random to the network. In our experiments, we set m equal to ten percent of the number of edges in each network to provide adequate anonymization, as suggested by Hay et al. [32]. Also, for the role-enhanced version, we vary the role equivalence threshold (δ) between 0.3 and 1, and average the measurements over 500 runs.

Table 11: Structural Properties of Social Network Datasets

Dataset	# Nodes	# Edges	Density	Diameter	APL	C.Coef.
PolBooks	105	441	0.081	7	3.079	0.348
Jazz	198	5484	0.281	6	2.235	0.520
ER	100	1000	0.202	3	1.813	0.203
BA	100	990	0.200	2	1.927	0.118
SW	100	1014	0.205	8	3.576	0.646

Both the Greedy-Swap and Supergraph algorithms ensure degree k-anonymity for network nodes. They begin by constructing a k-anonymous *degree sequence*. The Supergraph algorithm adds edges to the network until the network meets the anonymized degree sequence. The Greedy-Swap algorithm builds an anonymized random graph based on the degree sequence and swaps its edges to obtain a network close to the original network. More details on these algorithms can be found in Section 2.3.2.

In our experiments, we vary anonymization value k between 2 and 10 for role-enhanced Greedy-Swap, and evaluate edge-betweenness-enhanced Supergraph for $k = 10$ and by varying threshold β between 0.1 and 1. The reported measures are averaged based on 50 runs. Since our implementation of Supergraph only considers adding edges to the graph, i.e., increasing node degrees, it will not perform well for networks which have very few nodes with very high degrees, such as scale-free networks. Therefore, we do not perform the experiments on the Jazz and BA datasets in the case of the edge-betweenness-enhanced Supergraph algorithm. We found the above-mentioned number of runs good enough to represent performance results while accounting for existing randomness in the algorithms. Our efforts to include other edge-perturbation algorithms and asses our proposed scheme on them were not successful, because of either flaws with the techniques (e.g., clustering-based methods proposed in [76] as described in Section 2.3.2), or lack of available algorithmic details to implement them (e.g., [83]).

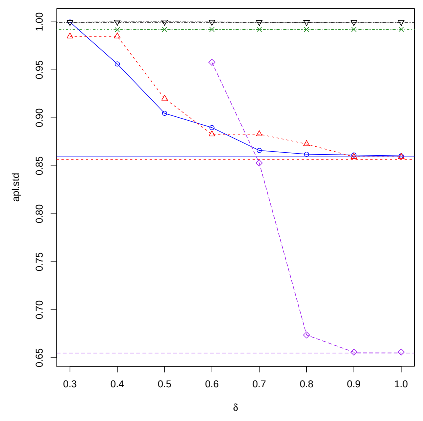
4.5.2 Experimental Results

In this section, we report on our experimental results based on the setup explained in Section 4.5.1. Figures 16, 17, and 18, respectively, demonstrate the performance of role-enhanced random perturbation, role-enhanced `Greedy-Swap`, and edge-betweenness-enhanced `Seupergraph` algorithms, in terms of preserving structural properties of original networks.

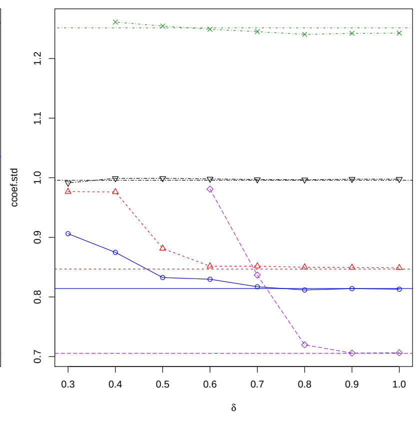
In all the figures, lines with symbols represent measurements corresponding to the enhanced algorithms while lines without symbols represent results of the corresponding original algorithms. The measurements are normalized based on the original networks' measurements. For instance, in the APL plots, we divide measurements for the `PolBooks` network by 3.079 which is the APL of the original network. Given that the measurements are normalized, the closer a measurement is to unity, the better the network structural properties has been preserved. Improvements over the original anonymization algorithms can be easily evaluated by subtracting the corresponding measurement differences to unity. For instance, if for a specific measure the original algorithm achieves 0.7 and the enhanced algorithm achieves 1.1, the improvement is calculated as $|1 - 0.7| - |1 - 1.1| = 0.2$. Due to brevity, we present only accuracy measurements for betweenness centrality. Closeness centrality results were very similar in nature to those of betweenness.

We summarize our findings based on the evaluation measurements reported in Figures 16, 17, and 18 in the following.

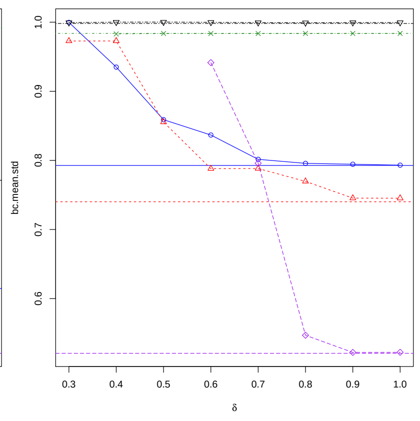
Dependency on dataset: As expected, the amount of distortion to the measures and effectiveness of our enhancement approach is not the same for different datasets. While the enhanced algorithms show better performance than their original counterparts in most cases, there are also cases with neutral and negligible negative effects. The role-enhanced random perturbation (Figure 16) has almost no effect on the random and scale-free datasets, while it has significant improvement for the other three networks, i.e., `PolBooks`, `Jazz`, and `SW`. The improvement can be attributed to their similar topological characteristics as in small-world networks. In the case of role-enhanced `Greedy-Swap` (Figure 17), datasets show moderate improvements on measures, with the exceptions of scale-fee dataset for the first three measures and small-world dataset for the accuracy measures. The edge-betweenness-enhanced



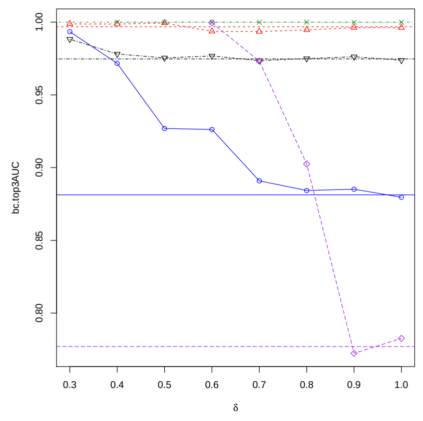
(a) Average Path Length



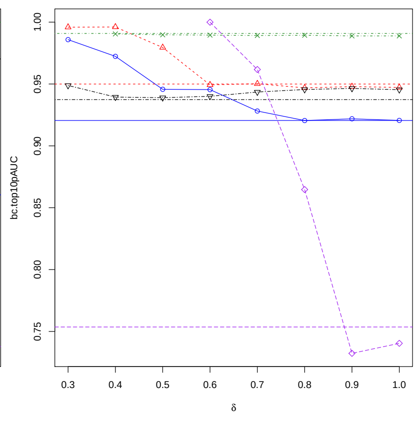
(b) Clustering Coefficient



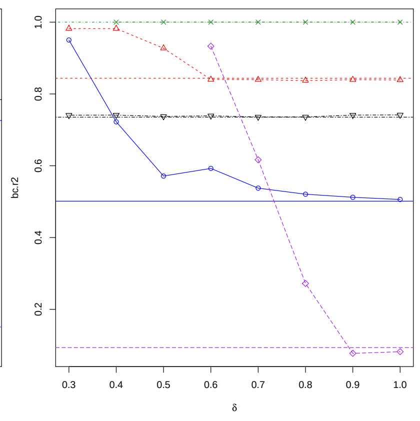
(c) Mean Betweenness Centrality



(d) BC *Top3AUC* Accuracy



(e) BC *Top10pAUC* Accuracy



(f) BC R^2 Accuracy

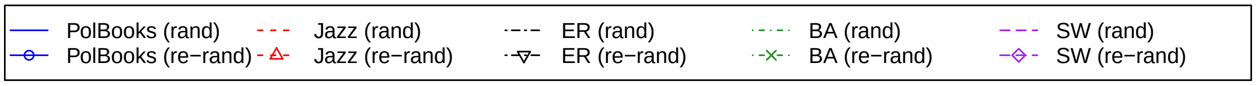
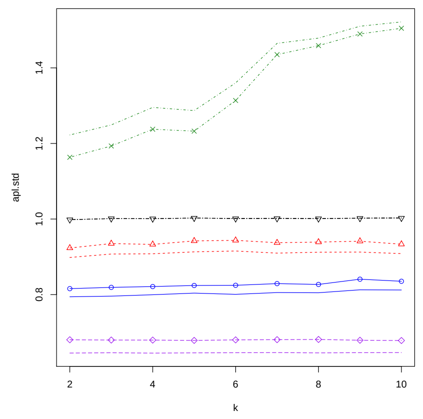
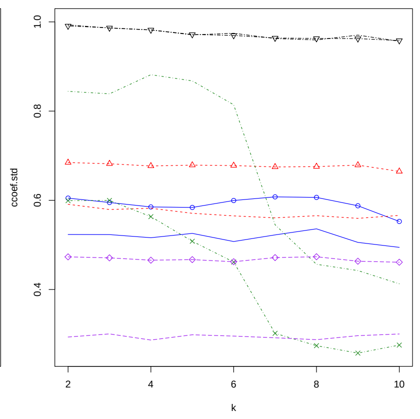


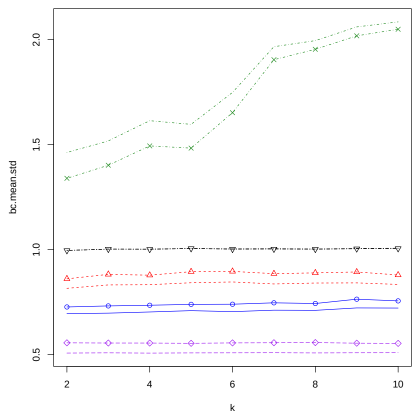
Figure 16: Evaluation Measurements for Role-Enhanced Random Perturbation ($m = 10\%|E|$, $0.3 \leq \delta \leq 1$)



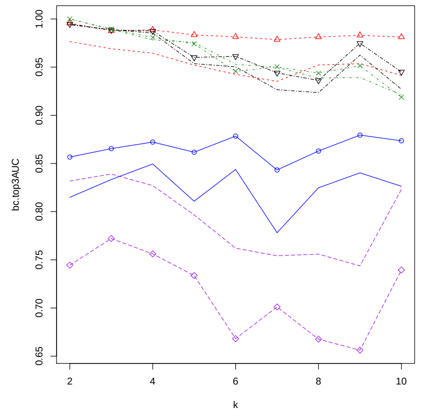
(a) Average Path Length



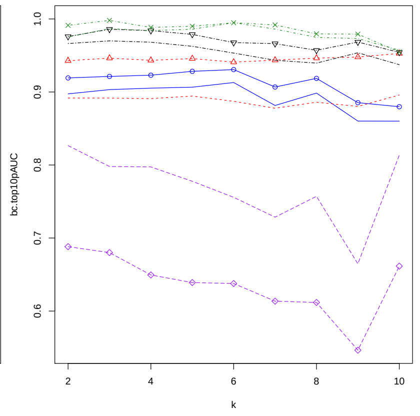
(b) Clustering Coefficient



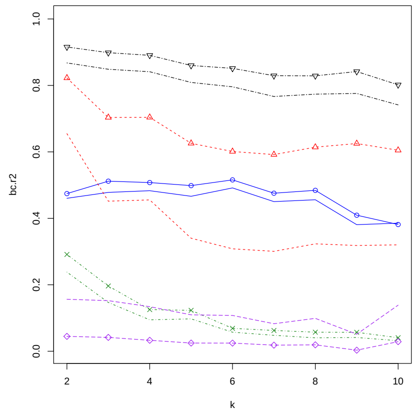
(c) Mean Betweenness Centrality



(d) BC *Top3AUC* Accuracy



(e) BC *Top10pAUC* Accuracy



(f) BC R^2 Accuracy

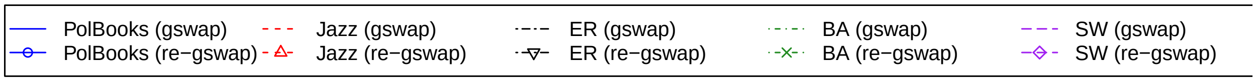
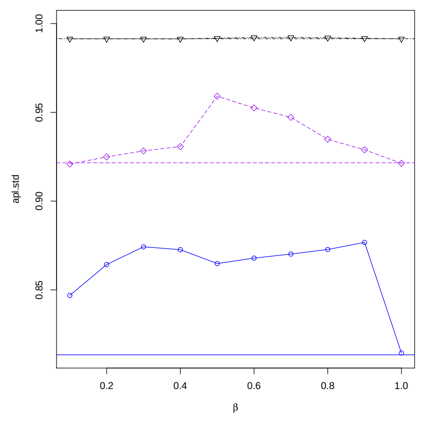
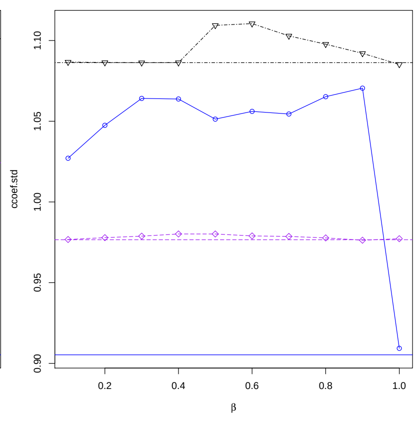


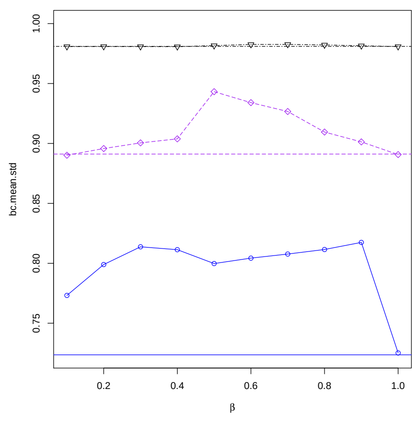
Figure 17: Evaluation Measurements for Role-Enhanced `Greedy-Swap` $(2 \leq k \leq 10)$



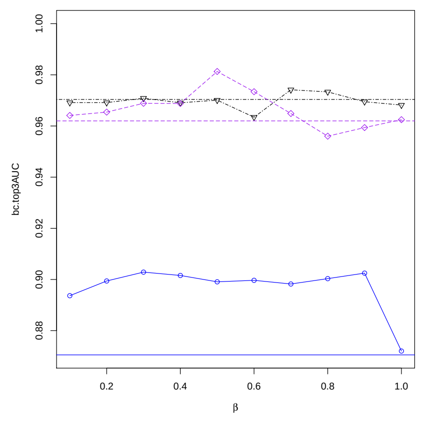
(a) Average Path Length



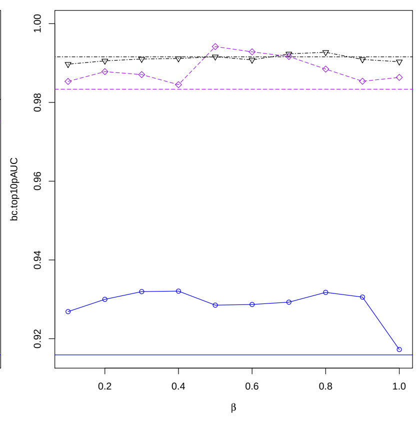
(b) Clustering Coefficient



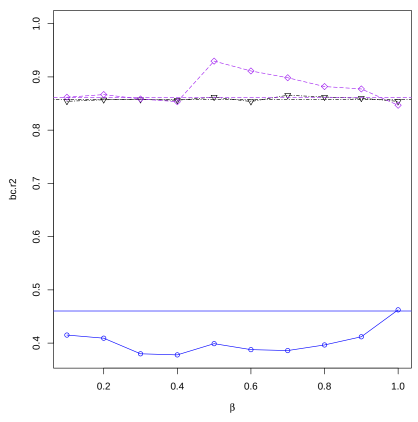
(c) Mean Betweenness Centrality



(d) BC *Top3AUC* Accuracy



(e) BC *Top10pAUC* Accuracy



(f) BC R^2 Accuracy

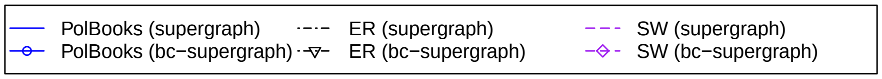


Figure 18: Evaluation Measurements for Edge-Betweenness-Enhanced `Supergraph` ($k = 10$, $0.1 \leq \beta \leq 1$)

`Supergraph` (Figure 18) also results in better performance in almost all the measures.

It is worth mentioning that the random graph (`ER`) is in fact very structurally robust to all the perturbation algorithms. Since the edges are randomly distributed in such a topology anyway, their replacements would not have much impact on the structural properties.

Effects of parameter variations: Improvements in the role-enhanced random perturbation (Figure 16) are negatively correlated with threshold δ. That is expected since increasing δ allows for less perfect role matching, and subsequently less structural preservation. Interestingly, *improvements on structural preservation is almost independent of anonymization parameter k.* This is suggested by the almost parallel lines in Figure 17 that correspond to the enhanced and original algorithms for each dataset. As for β, the edge-betweenness enhancement parameter, it does not show monotonic improvement as seen for δ. Strangely, the lower-range values of β perform worse than middle-range values in preserving the measures.

Measure correlations: We notice that the APL and mean BC plots follow similar overall pattern for the same networks, for all the experiments. Also, the mean CC plots follow the same pattern in a vertically mirrored fashion, which are omitted due to space limitations. However, note that the pattern similarity is not always perfect and the improvements are in different scales. Nevertheless, this observation suggests that APL, mean BC, and mean CC measures are highly correlated, independent of the change in anonymization parameters and enhancement thresholds. Such correlation can significantly help in choosing an appropriate anonymization enhancement method for a dataset since there will be less number of measures to be analyzed for preservation.

4.5.3 Selecting Appropriate Thresholds

As expected and also suggested by the experimental results, the improvement provided by our enhanced algorithms for iterative edge perturbation can be tuned based on threshold parameters. The role-enhanced and edge-betweenness-enhanced algorithms rely, respectively, on δ, the role dissimilarity threshold of considering nodes semi-equivalent, and β, the edge betweenness threshold to consider an edge for addition/removal. Intuitively, we should use a threshold that results in the best preservation of structural properties of a social network.

We can evaluate such preservation using different measures as suggested in our experiments. Based on our experiments, the amount of preservation depends on both the social network dataset and the threshold, and does not always follow a monotonic growth/decrease with threshold values. For instance, in the role-enhanced random perturbation results, lower δ values dominantly show better performance. However, choosing $\delta = 0.5$ over $\delta = 0.7$ for the `PolBooks` network provides less *Top1* BC centrality accuracy while the other BC accuracy values are almost the same. Considering no significant improvement in the first case in other measures as well, one may prefer $\delta = 0.7$ over $\delta = 0.5$. This simple example shows that selecting a proper threshold value is probably not possible before actually performing enhanced anonymization on the input network for different threshold values. Note that the anonymization is an offline process before publishing a social network dataset, and it seems feasible to invest some computation by trying out different thresholds to fine-tune anonymization which both guarantees anonymity criteria and provides usable output for analysis purpose.

4.6 PRIVACY ANALYSIS

In this section, we consider the effects of the proposed enhancement approaches on the anonymity of the edge-perturbation schemes.

4.6.1 Effect on Anonymity Property

The enhanced algorithms modify the original algorithms in a way that the anonymization criteria of the scheme is preserved. The enhanced versions of the iterative edge perturbation approach continue the iteration until they achieve the anonymity property, similar to the original algorithm (Compare Algorithms 4 and 5 vs. Algorithm 1). The same condition holds for the enhanced `Greedy-Swap` since it only modifies the gin calculation in the original algorithm (see Section 4.3.2).

Among the schemes discussed in our work, the degree k-anonymity-based schemes such as Supergraph and Greedy-Swap will completely fulfill their anonymity properties in the enhanced versions. Because the anonymity is provided based on achieving a node measure (here, degree) that is achieved in the enhanced versions too. Therefore, there is no degradation in provision of the original anonymity property. For instance, using role-enhanced Greedy-Swap, for any node in the perturbed graph, there will be at least $k-1$ other nodes with the same degree. So an adversary cannot re-identify nodes based on their degree. In the case of random perturbation, our enhanced algorithm still respects the anonymization criteria, i.e., adding and removing certain number of random edges. However, in terms of theoretical anonymity property, the enhanced algorithm will be slightly different from the original algorithm. Random perturbation creates uncertainty about the true network by creating multiple possible worlds. Since our enhancement approach eliminates some of the random choices it will slightly reduce the space of possible worlds. However, choosing a combination of large enough number of edge additions/removals and not-too-small threshold parameter in the enhanced algorithm can provide an acceptable degree of anonymity.

4.6.2 Role Structure as Background Knowledge

One plausible attack against role-enhanced perturbation could be to use role structure in the original and perturbed networks to re-identify nodes. Assume an adversary knows, for a target node t, role dissimilarities with every other node in the original network. We show them as a dissimilarity vector d_t of size $n = |V|$. The adversary can calculate role dissimilarities on the perturbed network, and then calculate correlation of d_t with that of every node, in the perturbed network, say d'_x. The nodes that have a high correlation in terms of role dissimilarities can be a potential match. However, such an attack is not computationally feasible. Since the attacker does not know the node identifiers, simply correlating the two vectors d_t and d'_x is not meaningful: a dissimilarity value at a certain index in d_t may not correspond to the dissimilarity value at the same index in d'_x. In fact, d_t should be correlated against all permutations of values in d'_x. So the computation itself is of complexity $O((n+1)!)$. Even if the attacker can perform all the computations, due to non-perfect role matching in our enhanced algorithm and high sensitivity of role dissimilarity calculation algorithm to small changes in the network structure, there is no guarantee that high correlation in role dissimilarities can help in node re-identification.

4.6.3 Preserved Measures as Background Knowledge

One may argue that if certain node measures such as betweenness centrality are preserved better using our enhancement approaches, they might be misused by an attacker for node re-identification. For instance, if an enhanced algorithm provides perfect *Top1AUC* BC accuracy, an attacker that knows about the most central node will be able to easily re-identify that node in the perturbed network. Although it seems infeasible as a practical attack, one can adjust anonymization parameters so as to introduce more distortion of the centrality measures for the anonymized network (e.g., choosing a larger δ value in the case of role-enhanced random perturbation). Note that assuming such information as attacker's background knowledge contradicts the goal of the enhanced algorithms, i.e., preserving the measures.

5.0 CONCLUSIONS AND DISCUSSIONS

In this chapter, we provide a summary of contributions in this dissertation work and discuss how they address the objectives that were proposed in Section 1.2 as well as their implications and limitations. Furthermore, we propose some future research directions based on approaches proposed in the dissertation.

5.1 SUMMARY OF CONTRIBUTIONS AND RESULTS

In this dissertation, we framed and addressed several challenges regarding user privacy in SNSs. We approached the privacy problem from both online and offline aspects of accessing privacy-sensitive data, i.e., whether data is accessed online within an SNS (by users through SNS functions) or offline where it is anonymized and exported for third party usage. From online protection perspective, we have proposed an ontology-based approach to capture, enforce, and analyze fine-grained privacy control policies in SNSs. In our proposed Ontology-Based SNS Access Control model (OSNAC), we have shown how fine-grained knowledge in an SNS can be captured using OWL language, and how such representation can be employed for fine-grained specification of protected resources and access subjects. By leveraging SWRL rule language, we proposed extensive sets of privacy control policy rules for both users and an SNS provider to enable flexible privacy control policy management in such an environment. We also demonstrated the suitability of such an ontology-based representation for enabling advanced policy analysis tasks in SNSs. From offline protection perspective, we have proposed novel approaches to preserve utility of social network data during anonymization in terms of preserving structural properties of social network datasets.

We have employed concepts from social network analysis theory to develop a general framework to improve k-anonymization algorithms in the literature from utility perspective while maintaining their privacy guarantees.

In the rest of this section, we discuss our contributions with regards to validating the main hypotheses of our work, and, particularly, to answering the more specific questions that we proposed in Section 1.2.

Question 1. Does OSNAC enable finer-grained specification of protected resources compared to the existing access control models for SNSs?

Our evaluation of OSNAC's expressiveness, presented in Section 3.2.5.1, provides confirmative evidence for the above question. We showed the support of common schemes of specifying protected resources in access control models by OSNAC including use of identifiers, attributes, and hierarchical organization of protected resources. Furthermore, by enumerating a set of flexible features suitable for SNS environments, we compared their support in OSNAC versus state of the art general-purpose access control models or those that have been specifically proposed for SNSs and Semantic Web. Those features include networked data, controlling semantics, hierarchies, policy authorship by both system and users, multiple authorities, and negative authorization and conflict resolution. While OSNAC supports all the above-mentioned features, other models provide either partial or no support in most cases. The related models compare as follows with regards to the first three of those factors that directly relate to fine-grained specification of protected resources. *Networked data* is leveraged fully by the Semantic Web-based access control framework for SNSs [13], but only partially by relationship-based access control models for SNSs [14, 15, 23, 3, 24, 25] and the access control model for RDF stores [66]. *Controlling semantics* is only supported by the access control model for RDF stores. And *hierarchies* are not supported by either relationship-based access control models or XACML [2]. As for policy authorship by both users and system and multiple authorities features, none of the related models provide sufficient support. Moreover, negative authorization is only supported flexibly by XACML.

Question 2. Does OSNAC enforce privacy control policies correctly?

We have shown that the access authorization procedure in OSNAC is correctly enforced

(Theorem 1). OSNAC supports both users and system policies and there exists various types of policy rules for each. We provided a detailed description of access entailment procedure in such a policy structure in Section 3.2.4, including the conflict resolution process based on the *deny-overrides* strategy. We have also empirically tested a prototype of an OSNAC access control engine, manually evaluating the correctness of access decisions in a small scenario. Moreover, our prototype implementation entails access decision much faster than its predecessor [56], making it more plausible for use in practice.

Question 3. Can our ontology-based privacy control policy analysis framework enable novel and useful policy analysis tasks for SNSs?

We have proposed a generic framework for formal ontology-based privacy control policy analysis, and have provided various example scenarios on how such a framework is enabling novel policy analysis tasks in SNSs in Section 3.3. Our detailed analysis task of verifying completeness of privacy control permissions, including verifiable properties and the conducted case study on Facebook, demonstrates a unique example of a policy analysis task that is vital for ensuring privacy in SNSs. However, such an analysis cannot be supported by other rather-powerful policy analysis frameworks such as EXAM [52].

Question 4. Do role-enhanced and edge-betweenness-enhanced versions of edge perturbing anonymization algorithms preserve data utility better than original algorithms?

Our experimental results on evaluating the effect of employing our framework on three of the prominent k-anonymization algorithms in the literature shows very promising improvements in preserving structural properties. In order to asses the generality of our framework, we have conducted experiments on several datasets with various social network topolgies and evaluated the anonymization results based on several network/node social network analysis measures. The results, which were presented in detail in Section 4.5.2, show overall positive effect of using our role-enhanced and edge-betweenness-enhanced algorithms in preserving various measures, i.e., resulting in closer numbers to the original datasets than original algorithms. A limitation of our utility-preserving framework is that the improvements are not guaranteed in a unified manner: not all measures achieve significant improvements and a very small number of measures show slightly negative effect. However, as it was the purpose of our extensive experimental results, they show overall positive achievement in various topology/measure situations.

Question 5. Do role-enhanced and edge-betweenness-enhanced versions of edge perturbing anonymization algorithms respect privacy requirements of the original algorithms?

We have shown in Section 4.6 that our utility-preserving anonymization framework will guarantee the anonymity properties of the k-anonymity-based schemes such as `Supergraph` and `Greedy-Swap`. This is due to achieving the same *anonymity* node measure in the enhanced versions of the algorithms compared to their corresponding original versions. The situation is less perfect with regards to the random perturbation algorithm. Since random perturbation uniformly increases the uncertainty about the true links in the social network, any restriction on edge addition/removal will change such uniform distribution of uncertainty. However, increasing the number of links to add/remove from the social network can compensate for such degradation in privacy due to not-perfectly-uniform uncertainty introduced by our enhanced algorithm. We also argue that using the role structure against the role-enhancement procedure is not a feasible threat; it is infeasible to align nodes in two networks by calculating role dissimilarity in the anonymized version. Even if such an alignment was feasible, the non-perfect role matching in our scheme prevents an attacker from calculating role dissimilarity measures on the anonymized network that are close to the ones of the original network. Furthermore, use of preserved measures by the framework as background knowledge by an attacker can be prevented by adjusting the thresholds in the enhancement algorithms (e.g., role dissimilarity threshold). However, note that considering such a threat is in conflict with the consideration of using the framework for preserving the measures in the first place.

5.2 LIMITATIONS

There are some limitations with regards to the research conducted in this dissertation which we discuss below.

Usability is an important aspect of security and privacy solutions [17], and it becomes a primary concern when lay users of SNSs need to specify privacy control policies. The scope of our work with regards to OSNAC is limited to providing a fine-grained and powerful policy specification and enforcement framework, as a foundation layer for privacy control in SNSs. We consider that an SNS operator will need to build layers on top of our framework that will allow users to specify policies in a user-friendly manner. Therefore, we have not conducted usability testings for the current framework. Ensuring that privacy control policies are understandable by both system administrators and users needs further usability studies.

In experimenting with our OSNAC engine prototype, we limit our tests to relatively simple access scenarios in order to avoid accounting for numerous variables that could change with regards to the dataset and policy rules. A limitation of such an approach is that we do not assess the effect of complexities of policies on the performance of the framework. Designing such an experiment requires a method to characterize the complexity of policies. That itself can be a quite complex task, considering the various influencing factors including the numbers and formats of predicates that are used in the rules, the format of the rules that can interact with each other, etc.

In terms of dynamicity of the information in an SNS, since OSNAC enforces its policy rules based on an SNS knowledge base, any change in the knowledge base will be automatically reflected in the enforcement of privacy control policies. However, the scope of our policy analysis framework is limited to the changes that might occur in the privacy control policies. It would be interesting to explore dynamicity of SNS knowledge as a factor in policy analysis.

Our role-enhanced and edge-betweenness-enhanced approaches rely on algorithms with complexities $O(n^3)$ and $O(nm)$, respectively, which may sound limiting in terms of scalability to large graphs. However, there are several considerations. Anonymization is an offline process. Therefore, a heavy but feasible amount of computation would be reasonable in practice. A data analyst may typically use algorithms such as community detection that require similar order of complexity computation. Therefore, it will be worthwhile to invest that amount of computation on generating a quality anonymized dataset in the first place. Furthermore, we can employ approximate algorithms [68] and parallel processing approaches

[5] to improve metric calculation.

As discussed in Section 5.1, our enhanced version of random perturbation algorithm does not perfectly match the privacy guarantee of the original algorithm. This is due to potential rejection of add/removal of some of the randomly selected edges by the enhanced algorithm. We have not quantified such diversion from uniform randomness in the dissertation. Moreover, although we have shown through extensive experiments that enhanced algorithms preserve data utility, we have not established theoretical proofs that they do so.

5.3 FUTURE RESEARCH

There are several areas for future research in the context of this dissertation. One of the ultimate goals of relying on Semantic Web technologies for OSNAC is to enable easier adoption of such a model for privacy protection in real-world systems. Our evaluation of an OSNAC-based prototype engine required few millisecond to infer access decision for relatively small systems (less than 10,000 users). Our current implementation clearly does not scale for today's massive SNSs like Facebook with billions of users. Further research is required to design and implement access control engines based on scalable Semantic Web tools, which may involve redesigning the policy model too. Another future research with regards to OSNAC is to introduce the signature characteristics of OSNAC, i.e., ontology-based policy specification and multiple policy authors, into standard policy languages such as XACML. Such an approach can improve adoptability of the model in practice and its extensibility given the body of literature about specification and enforcement of such languages.

In terms of analyzing privacy control policies, we have focused on developing a theoretical framework that enables ontology-based policy analysis. Since many analysis scenarios in SNSs would involve user policies, one of the foreseeable extensions to this work is to develop policy analysis tools that collect and inspect user policies in SNSs automatically and to study the impact of such tools on improving privacy issues. A specific example with regards to our completeness analysis of privacy control permissions is to develop a tool that can identify and extract missing policies automatically. In our current prototype, we are able to verify the privacy properties and manually verify crafted missing permissions as detailed in Section 3.3.3.2. A fully automatic extraction mechanism of such missing permissions requires extending an ontology reasoning engine to be able to extract meaningful subsumed concepts according to the ontology model at hand.

We have shown through an extensive empirical study how social network analysis concepts such as structural roles can be leveraged to anonymize social network data more efficiently with regards to data utility. An open research problem is to characterize and prove such an approach mathematically, e.g., by providing lower bounds on improvements in preserving data utility. Proposing enhancement approaches with better algorithmic complexity that will be suitable for large graphs is another potential future work. In addition to employing faster algorithms for metric calculation [5, 68], it is interesting to explore use of more local structural metrics that would require less intense computation for large graphs. Moreover, in this work, we focused on a specific class of privacy enhancing techniques, i.e., anonymization. It will be interesting to explore use of similar techniques in conjunction with other privacy preserving approaches of sharing information such as differential privacy [22].

BIBLIOGRAPHY

[1] Electronic Privacy Information Center - Facebook Privacy. http://epic.org/privacy/facebook/.

[2] eXtensible Access Control Markup Language (XACML) Version 2.0. Technical report, OASIS, 2005.

[3] M. Anwar, Z. Zhao, and P. W. L. Fong. An access control model for Facebook-style social network systems. Technical report, Department of Computer Science, University of Calgary, July 2010.

[4] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *Proc. 16th Int'l conference on World Wide Web*, pages 181–190, Banff, Alberta, Canada, 2007. ACM.

[5] D. A. Bader and K. Madduri. SNAP, Small-world Network Analysis and Partitioning: an open-source parallel graph framework for the exploration of large-scale networks. In *Proc. IEEE International Symposium on Parallel and Distributed Processing*, 2008.

[6] E. Barka and R. S. Sandhu. Framework for Role-based Delegation Models. In *Proc. 16th Annual Computer Security Applications Conference*, pages 168–176. IEEE Computer Society, Dec. 2000.

[7] E. Bertino, P. Samarati, and S. Jajodia. Authorizations in relational database management systems. In *Proceedings of the 1st ACM conference on Computer and communications security - CCS '93*, pages 130–139, New York, New York, USA, Dec. 1993. ACM Press.

[8] S. P. Borgatti, K. M. Carley, and D. Krackhardt. On the robustness of centrality measures under conditions of imperfect data. *Social Networks*, 28(2):124–136, May 2006.

[9] S. P. Borgatti and M. G. Everett. Two algorithms for computing regular equivalence. *Social Networks*, 15(4):361–376, Dec. 1993.

[10] D. Boyd and E. Hargittai. Facebook privacy settings: Who cares? *First Monday*, 15(8), July 2010.

[11] Brian Hayes. Connecting the Dots: Can the tools of graph theory and social-network studies unravel the next big plot? *American Scientist*, 94(5), 2006.

[12] A. Campan and T. M. Truta. A clustering approach for data and structural anonymity in social networks. In *Proc. 2nd ACM SIGKDD Int'l Workshop on Privacy, Security, and Trust in KDD (PinKDD'08)*, 2008.

[13] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham. A semantic web based framework for social network access control. In *Proc. 14th ACM Symposium on Access Control Models and Technologies*, pages 177–186. ACM, 2009.

[14] B. Carminati, E. Ferrari, and A. Perego. Rule-Based Access Control for Social Networks. In R. Meersman, Z. Tari, and P. Herrero, editors, *Proc. OTM 2006 Workshops (On the Move to Meaningful Internet Systems)*, volume 4278 of *LNCS*, pages 1734–1744. Springer, Oct. 2006.

[15] B. Carminati, E. Ferrari, and A. Perego. Enforcing access control in Web-based social networks. *ACM Trans. Inf. Syst. Secur.*, 13(1):1–38, Nov. 2009.

[16] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, and P. Samarati. k-Anonymous data mining: a survey. In *Advances in Database Systems*, chapter 1, pages 105–136. Springer, 2008.

[17] L. Cranor and S. Garfinkel. *Security and Usability.* O'Reilly Media, Inc., 2005.

[18] E. Damiani, P. Milano, S. D. di Vimercati, S. Paraboschi, and P. Samarati. A fine-grained access control system for XML documents. *ACM Transaction on Information Systems and Security*, 5(2):169–202, 2002.

[19] A. Dersingh, R. Liscano, A. Jost, J. Finnson, and R. Senthilnathan. Utilizing Semantic Knowledge for Access Control in Pervasive and Ubiquitous Systems. *Mobile Networks and Applications.*

[20] S. Dietzold and S. Auer. Access control on RDF triple stores from a semantic wiki perspective. 2006.

[21] N. Dunlop, J. Indulska, and K. Raymond. Methods for Conflict Resolution in Policy-Based Management Systems. In *Proc. 7th IEEE Int'l Enterprise Distributed Object Computing Conference*, pages 98–109. IEEE Computer Society, 2003.

[22] C. Dwork. Differential privacy: A survey of results. *Theory and Applications of Models of Computation*, pages 1–19, 2008.

[23] P. Fong, M. Anwar, and Z. Zhao. A Privacy Preservation Model for Facebook-Style Social Network Systems. In M. Backes and P. Ning, editors, *Computer Security ESORICS 2009*, volume 5789 of *Lecture Notes in Computer Science*, book part (with own title) 19, pages 303–320. Springer Berlin / Heidelberg, 2009.

[24] P. W. Fong. Relationship-based access control: protection model and policy language. CODASPY '11, pages 191–202, San Antonio, TX, USA, 2011. ACM.

[25] P. W. Fong and I. Siahaan. Relationship-based access control policies and their policy languages. In *Proc. 16th ACM Symposium on Access Control Models and Technologies*, SACMAT '11, pages 51–60, Innsbruck, Austria, 2011. ACM.

[26] T. Frantz, M. Cataldo, and K. Carley. Robustness of centrality measures under uncertainty: Examining the role of network topology. *Computational & Mathematical Organization Theory*, 15(4):303–328, Dec. 2009.

[27] P. Gleiser and L. Danon. Community Structure in Jazz. *Advances in Complex Systems*, 6(4):565–573, 2003.

[28] J. Golbeck. *Computing and applying trust in web-based social networks*. PhD thesis, University of Maryland, 2005.

[29] J. Golbeck and J. Hendler. Inferring binary trust relationships in Web-based social networks. *ACM Transactions on Internet Technology*, 6(4):497–529, 2006.

[30] R. Gross, A. Acquisti, and H. J. Heinz. Information revelation and privacy in online social networks. In V. Atluri, S. D. di Vimercati, and R. Dingledine, editors, *Proc. ACM Workshop on Privacy in the Electronic Society*, pages 71–80. ACM, Nov. 2005.

[31] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis. Resisting structural re-identification in anonymized social networks. *Proc. VLDB Endow.*, 1(1):102–114, 2008.

[32] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava. Anonymizing Social Networks. Technical Report 07-19, University of Massachusetts Amherst, 2007.

[33] X. He, J. Vaidya, B. Shafiq, N. Adam, and V. Atluri. Preserving Privacy in Social Networks: A Structure-Aware Approach. In *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT '09. IEEE/WIC/ACM International Joint Conferences on*, volume 1, pages 647–654. IEEE Computer Society, 2009.

[34] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph. OWL 2 Web Ontology Language - Primer. http://www.w3.org/TR/owl2-primer/, 2009.

[35] C. M. Hoadley, H. Xu, J. J. Lee, and M. B. Rosson. Privacy as information access and illusory control: The case of the Facebook News Feed privacy outcry. *Electronic Commerce Research and Applications*, 9(1):50–60, Jan. 2010.

[36] M. Horridge and P. F. Patel-Schneider. OWL 2 Web Ontology Language - Manchester Syntax. http://www.w3.org/TR/owl2-manchester-syntax/, 2012.

[37] I. Horrocks, P. F. Patel Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language - Combining OWL and RuleML. http://www.w3.org/Submission/SWRL/, 2004.

[38] H. Hu and G.-j. Ahn. Multiparty Authorization Framework for Data Sharing in Online Social Networks. In Y. Li, editor, *Proceedings of the 25th annual IFIP WG 11.3 conference on Data and applications security and privacy*, volume 6818 of *Lecture Notes in Computer Science*, pages 29–43. Springer Berlin / Heidelberg, 2011.

[39] A. Jain and C. Farkas. Secure resource description framework: an access control model. pages 121–129, Lake Tahoe, California, USA, 2006. ACM.

[40] A. Jø sang, R. Hayward, S. Pope, and A. Josang. Trust network analysis with subjective logic. In *Proc. 29th Australasian Computer Science Conference*, volume 48, pages 85–94, Darlinghurst, Australia, 2006. Australian Computer Society, Inc.

[41] A. Jø sang, R. Ismail, C. Boyd, and A. Josang. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, Mar. 2007.

[42] L. Kagal, T. Finin, M. Paolucci, N. Srinivasan, K. Sycara, and G. Denker. Authorization and privacy for semantic Web services. *IEEE Intelligent Systems*, 19(4):50–56, July 2004.

[43] A. Kamdar. Facebook Graph Search: Privacy Control You Still Don't Have. *Electronic Frontier Foundation*, Jan. 2013.

[44] B. Karrer, E. Levina, and M. E. Newman. Robustness of community structure in networks. *Physical Review E*, 77(4), 2008.

[45] D. Knoke and S. Yang. *Social Network Analysis (Quantitative Applications in the Social Sciences)*. Sage Publications, Inc, 2nd edition, Nov. 2008.

[46] V. Kolovski, J. Hendler, and B. Parsia. Analyzing web access control policies. In *Proc. 16th Int'l Conference on World Wide Web*, pages 677–686, 2007.

[47] B. Krishnamurthy and C. E. Wills. Characterizing privacy in online social networks. WOSP '08, pages 37–42, Seattle, WA, USA, 2008. ACM.

[48] S. R. Kruk. FOAF-Realm: control your friends access to the resource. 2004.

[49] S. R. Kruk, S. Grzonkowski, A. Gzella, T. Woroniecki, and H. C. Choi. D-FOAF: Distributed Identity Management with Access Rights Delegation. pages 140–154. Springer, 2006.

[50] J. Lerner. Role Assignments. In U. Brandes and T. Erlebach, editors, *Network Analysis*, volume 3418 of *Lecture Notes in Computer Science*, book chapter/section 9, pages 216–252. Springer Berlin / Heidelberg, 2005.

[51] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining - KDD '05*, page 177, New York, New York, USA, Aug. 2005. ACM Press.

[52] D. Lin, P. Rao, E. Bertino, N. Li, and J. Lobo. EXAM: a comprehensive environment for the analysis of access control policies. *International Journal of Information Security*, 9(4):253–273, Aug. 2010.

[53] K. Liu and E. Terzi. Towards identity anonymization on graphs. In *Proc. ACM SIGMOD Int'l Conference on Management of Data*, pages 93–106, Vancouver, Canada, 2008. ACM.

[54] M. Liu, D. Xie, P. Li, X. Zhang, and C. Tang. Semantic Access Control for Web Services. In *Networks Security, Wireless Communications and Trusted Computing, 2009. NSWCTC '09. International Conference on*, volume 2, pages 55–58, 2009.

[55] E. Lupu and M. Sloman. Conflict Analysis for Management Policies. *Policy*, 97(May):1–14, 1997.

[56] A. Masoumzadeh and J. Joshi. Ontology-based access control for social network systems. *International Journal of Information Privacy, Security and Integrity (Special Issue: Selected Papers from PASSAT 2010)*, 1(1):59–78, Jan. 2011.

[57] D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language - Overview. http://www.w3.org/TR/owl-features/, 2004.

[58] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement - IMC '07*, page 29, New York, New York, USA, Oct. 2007. ACM Press.

[59] J. Moffett and M. Sloman. Policy Conflict Analysis in Distributed System Management. *Journal of Organizational Computing*, 4(1):1–22, 1994.

[60] B. Motik, U. Sattler, and R. Studer. Query Answering for OWL-DL with rules. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(1):41–60, July 2005.

[61] A. A. Nanavati, S. Gurumurthy, G. Das, D. Chakraborty, K. Dasgupta, S. Mukherjea, and A. Joshi. On the structural properties of massive telecom call graphs. In *Proceedings of the 15th ACM international conference on Information and knowledge management - CIKM '06*, page 435, New York, New York, USA, Nov. 2006. ACM Press.

[62] A. Narayanan and V. Shmatikov. De-anonymizing Social Networks. In *Security and Privacy, IEEE Symposium on*, pages 173–187, Oakland, CA, USA, 2009. IEEE.

[63] M. Newman. *Networks: An Introduction*. Oxford University Press, 1 edition, May 2010.

[64] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2), 2004.

[65] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF, 2008.

[66] P. Reddivari, T. Finin, and A. Joshi. Policy-based access control for an RDF store. In *Proc. Policy Management for the Web Workshop*, pages 78–81, 2005.

[67] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '02*, page 61, New York, New York, USA, July 2002. ACM Press.

[68] M. Riondato and E. M. Kornaropoulos. Fast approximation of betweenness centrality through sampling. In *Proc. 7th ACM international conference on Web search and data mining*, pages 413–422, 2014.

[69] T. Ryutov, T. Kichkaylo, and R. Neches. Access Control Policies for Semantic Networks. In *Policies for Distributed Systems and Networks, 2009. POLICY 2009. IEEE International Symposium on*, pages 150–157, 2009.

[70] P. Samarati, S. de Vimercati, and S. D. Capitani. Access Control: Policies, Models, and Mechanisms. In R. Focardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design*, volume 2171 of *Lecture Notes in Computer Science*, book part (with own title) 3, pages 137–196. Springer Berlin Heidelberg, 2001.

[71] D. Schwartz. 8 Facebook privacy flaps. *CBC News*, Sept. 2012.

[72] A. C. Squicciarini, M. Shehab, and F. Paci. Collective privacy management in social networks. pages 521–530, Madrid, Spain, 2009. ACM.

[73] A. C. Squicciarini, M. Shehab, and J. Wede. Privacy policies for shared content in social network sites. *The VLDB Journal*, 19(6):777–796, Dec. 2010.

[74] V. S. Subrahmanian, S. Jajodia, P. Samarati, and M. L. Sapino. Flexible Support for Multiple Access Control Policies. *ACM Transactions on Database Systems*, 26(2):214–260, 2001.

[75] L. Sweeney. k-anonymity: a model for protecting privacy. *Int'l Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[76] B. Thompson and D. Yao. The union-split algorithm and cluster-based anonymization of social networks. In *Proc. 4th Int'l Symposium on Information, Computer, and Communications Security (ASIACCS '09)*, pages 218–227, Sydney, Australia, 2009. ACM.

[77] W. Villegas, B. Ali, and M. Maheswaran. An Access Control Scheme for Protecting Personal Data. pages 24–35, 2008.

[78] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications.* Cambridge University Press, 1994.

[79] X. Wu, X. Ying, K. Liu, and L. Chen. A Survey of Privacy-Preservation of Graphs and Social Networks. volume 40 of *Advances in Database Systems*, book part (with own title) 14, pages 421–453. Springer US, 2010.

[80] X. Ying, K. Pan, X. Wu, and L. Guo. Comparisons of randomization and K-degree anonymization schemes for privacy preserving social network publishing. In *Proc. 3rd SIGKDD Workshop on Social Network Mining and Analysis (SNA-KDD 09)*, Paris, France, 2009. ACM.

[81] E. Zheleva and L. Getoor. Preserving the Privacy of Sensitive Relationships in Graph Data. In F. Bonchi, E. Ferrari, B. Malin, and Y. Saygin, editors, *Proceedings of the International Workshop on Privacy, Security, and Trust in KDD (PinKDD'07)*, Lecture Notes in Computer Science, pages 153–171. Springer Berlin Heidelberg, 2008.

[82] B. Zhou. A Brief Survey on Anonymization Techniques for Privacy Preserving Publishing of Social Network Data. *Social Networks*, 2007.

[83] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *IEEE 24th Int'l Conference on Data Engineering (ICDE '08)*, volume 00, pages 506–515, Cancun, Mexico, 2008. IEEE.